PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-202409
(43)Date of publication of application : 09.08.1996

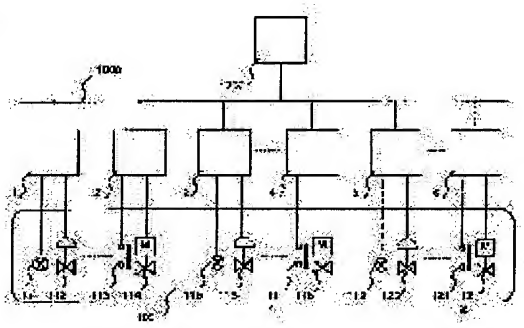
(51)Int.Cl. G05B 15/02
G06F 15/16
G06F 15/16

(21)Application number : 07-012293
(22)Date of filing : 30.01.1995

(71)Applicant : HITACHI LTD
(72)Inventor : YAMAOKA HIROMASA
WATABE MITSURU
KASAHARA TAKAYASU
SHIBATA KATSUNARI
MATSUBARA KIYOSHI
MOROOKA YASUO
FUNABASHI SEIJU

(54) DECENTRALIZED CONTROL UNIT, SYSTEM, AND CONTROLLER

(57)Abstract:
PURPOSE: To provide a controller which has many kinds of various control function for a decentralized control system.
CONSTITUTION: Controllers 1-6 consist of plural processors which perform numerical operation, sequence operation, neural net(NN) operation, fuzzy operation, etc. The controllers 1-6 are reduced in size by the configuring technology with LSI. Further, processor functions and I/O functions are made programmable to increase flexibility. A program incorporates a simple standard program and learnt models of the NN, etc., and weights and adds plural calculated outputs to generate output data. Further, the value of weighting is updated by learning. Consequently, the decentralized control system having many kinds of various control function such as numerical operation control, sequence control, NN application control, and fuzzy application control and the controllers used for them can be provided.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報**（A）　(11)特許出願公開番号

特開平8－202409

(43)公開日　平成８年(1996)８月９日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０５Ｂ　15/02				
Ｇ０６Ｆ　15/16	３７０　Ｚ			
	４７０　Ｂ			
		9063－3H	Ｇ０５Ｂ　15/ 02	Ｍ

審査請求　未請求　請求項の数21　ＯＬ　（全 50 頁）

(21)出願番号　特願平7－12293

(22)出願日　平成７年(1995)１月30日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　山岡　弘昌
茨城県日立市大みか町五丁目２番１号　株式会社日立製作所大みか工場内

(72)発明者　渡部　満
茨城県日立市大みか町七丁目１番１号　株式会社日立製作所日立研究所内

(72)発明者　笠原　孝保
茨城県日立市大みか町七丁目２番１号　株式会社日立製作所エネルギー研究所内

(74)代理人　弁理士　小川　勝男

最終頁に続く

(54)【発明の名称】　分散制御装置、システム及びコントローラ

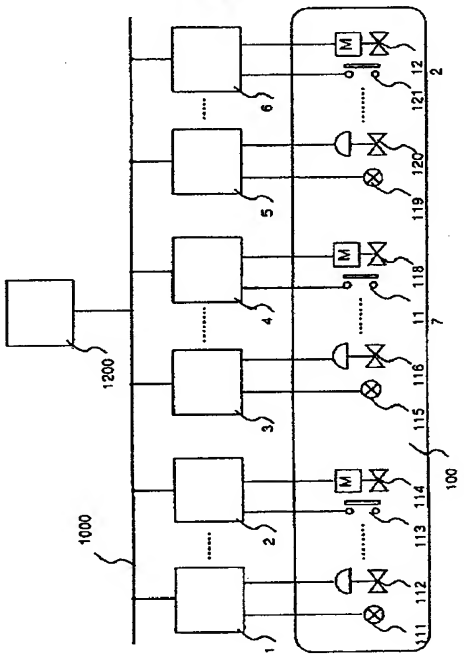
(57)【要約】

【目的】分散制御システムにおいて、多種多様な制御機能を有したコントローラを実現すること。

【構成】コントローラを数値演算，シーケンス演算，ニューラルネット（ＮＮ），ファジィなどを実行する複数のプロセッサで構成する。コントローラはＬＳＩ化技術により小型化する。さらに、プロセッサ機能、Ｉ／Ｏ機能をプログラマブルにすることにより柔軟性を増す。プログラムが簡単な標準プログラムとＮＮ等の学習モデルを組み込んで、計算される複数の出力を、重み付けして加算することによって出力データを生成する。かつ該重み付けの値を学習によって更新していく。

【効果】本発明により、数値演算制御，シーケンス制御，ＮＮ応用制御，ファジィ応用制御など多種多様な制御機能を有した分散制御システム及びそれに用いるコントローラを実現することができる。

図　1



1

【特許請求の範囲】

【請求項１】複数の制御対象機器を有するプロセスの制御装置において制御対象機器に制御指令を発するコントローラを各制御対象機器に対応して現場設置したことを特徴とする分散制御装置。

【請求項２】複数のプロセッサからなる制御システム用コントローラにおいて各コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、他のプロセッサが負荷の代替を行うことを特徴とするコントローラ。

【請求項３】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラは複数のプロセッサから構成され、該コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、同一コントローラ内の他のプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

【請求項４】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラはメモリまたは入出力を共有する複数のプロセッサから構成され、該伝送手段は該コントローラ内プロセッサ群の正常、異常に関係なく該コントローラの上記メモリまたは入出力の内容を読み書きできるように構成したことを特徴とする分散制御システム。

【請求項５】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、上記プロセッサ群は上記伝送手段を制御する通信プロセッサを含み、該通信プロセッサが故障時は他のプロセッサが通信プロセッサを代替するように構成したことを特徴とする分散制御システム。

【請求項６】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、上記プロセッサのうち一つのプロセッサが故障時は他のプロセッサが該故障プロセッサを代替するように構成したことを特徴とする分散制御システム。

【請求項７】複数のプロセッサからなる制御システム用コントローラにおいて、各プロセッサの故障を検出する第１の手段（６１８０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段（６１８０）が一つのプロセッサの故障を検出した場合に、第２の手段（６１６０）に記憶された各プロセッサの負荷状況に応じて第３の手段（６１７０）が故障したプロセッサの負荷を他の一つ以上のプロセッサに対して分割して代替させることを特徴とするコントローラ。

2

【請求項８】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第１の手段（６１８０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段(６１８０)が一つのプロセッサの故障を検出した場合に、第２の手段(６１６０)に記憶された各プロセッサの負荷状況に応じて第３の手段（６１７０）が故障したプロセッサのタスクを他の一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。

【請求項９】複数のプロセッサからなり、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第１の手段（６１８０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段(６１８０)が一つのプロセッサの故障を検出した場合に、第２の手段(６１６０)に記憶された各プロセッサの負荷状況に基づいて、第３の手段（６１７０）が故障したプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。

【請求項１０】複数のプロセッサを有する制御システム用コントローラにおいて、各プロセッサの負荷を検出する第１の手段（６１４０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段（６１４０）が一つのプロセッサの負荷オーバを検出した場合に、第２の手段（６１６０）に記憶された各プロセッサの負荷状況に応じて第３の手段（６１７０）が負荷オーバとなったプロセッサの負荷を一つ以上のプロセッサに対して分割して代替させることを特徴とするコントローラ。

【請求項１１】複数のプロセッサからなり、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第１の手段（６１４０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段（６１４０）が一つのプロセッサの負荷オーバを検出した場合に、第２の手段(６１６０)に記憶された各プロセッサの負荷状況に応じて第３の手段(６１７０)が負荷オーバとなったプロセッサのタスクを一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。

【請求項１２】複数のプロセッサを有し、各プロセッサ

が複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第１の手段（６１４０）と、各プロセッサの負荷状況を記憶する第２の手段（６１６０）と、各プロセッサの負荷を制御する第３の手段（６１７０）を有し、第１の手段（６１４０）が一つのプロセッサの負荷オーバを検出した場合に、第２の手段（６１６０）に記憶された各プロセッサの負荷状況に基づいて、第３の手段（６１７０）が負荷オーバとなったプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。

【請求項１３】複数のコントローラと各コントローラを接続するネットワークを有する分散制御システムにおいて、各コントローラは自コントローラの負荷を測定し測定した負荷をネットワークに送信する第１の手段（６１４０）と、任意のコントローラの負荷の代替を他のコントローラに依頼する第２の手段（６１２０）と、他のコントローラから代替の依頼に対して自己の負荷状況に基づき回答する手段をする第３の手段を有し、何れかのコントローラが負荷オーバあるいは故障した場合、その負荷オーバあるいは故障を検出したコントローラの第２の手段は各第１の手段から伝えられた負荷状況に応じて負荷の代替を依頼するコントローラを定め負荷の代替を依頼し、依頼されたコントローラの第３の手段は自己の負荷状況に応じて代替する負荷を定めこの負荷の実行を制御すると共にこの代替状況を回答することを特徴とした分散制御システム。

【請求項１４】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて少なくとも一台のコントローラはグローバルメモリまたは入出力を共有する複数のプロセッサから構成され、上記プロセッサはキャッシュメモリおよび該キャッシュメモリの内容とグローバルメモリの内容一致化手段を有し、グローバルメモリの内容または入出力の内容は上記伝送手段を介して他のコントローラから読み取り可能なように構成したことを特徴とする分散制御システム。

【請求項１５】複数のコントローラと各コントローラを接続する伝送手段からなる分散制御システムにおいて少なくとも一台以上のコントローラは入出力有し、該入出力の接続がプログラマブルであることを特徴とする分散制御システム。

【請求項１６】複数のコントローラを有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、該複数のプロセッサ各々は自己の健全／異常の判別手段、各プロセッサの負荷を測定する手段を有し、一つのプロセッサが負荷オーバーになると、他のプロセッサのうち最も負荷の小さいプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

【請求項１７】複数のプロセッサを有する制御システム用コントローラにおいて、該複数のプロセッサのうち少なくとも一つのプロセッサはニューロ演算，ファジィ演算，数値演算またはシーケンス演算を実行することを特徴とするコントローラ。

【請求項１８】それぞれが情報を格納するメモリを有する複数のプロセッサと上記各メモリ、各プロセッサ及び外部のネットワークとの間で上記メモリの情報を授受するために上記相互を連絡する伝送線と少なくとも一つのプロセッサにネットワークと各メモリ間の情報授受を制御するネットワーク制御機能を設定する手段とを有し、上記手段により特定された一つのプロセッサによって、上記ネットワークと上記メモリ間の情報授受を制御するようにしたことを特徴とするコントローラ。

【請求項１９】センサ等からの入力を用いて制御出力を求める制御システムにおいて、該入力から少なくとも１つのニューラルネットワークを含む複数の手段または方法によって計算される複数の出力を、重み付けして総合することによって制御出力を生成し、かつ該重み付けの値を学習によって更新していくことを特徴とする適応的制御システム。

【請求項２０】複数のコントローラをネットワークで接続して制御を実行する分散制御システムにおいて、該複数のコントローラにおいて感知した異常をネットワーク上に送信するか否かの判定をネットワークからの受信情報に基づいて各コントローラで実行する分散型コントローラの診断方式。

【請求項２１】それぞれの機能及び相互接続がプログラマブルなプロセッサを複数内蔵するシングルチップコントローラ。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は各種プロセス制御，プラント制御，車両制御等において複数の制御装置（コントローラ）を用いて分散制御を行う分散制御システムに関する。

【０００２】

【従来の技術】従来からプラント制御システムの例では、複数のコントローラを分散して制御を行う分散制御システム、ＤＣＳ（Distributed Control System）と呼ばれる制御システムがある。このＤＣＳは複数のコントローラとマンマシンシステムとしてのオペレーターズコンソールをネットワークで結びプラントを制御しようとするものである。しかし、分散システムといってもコントローラ自体は計器室、電気室あるいは計算機室等と呼ばれる専用の設置スペースに集中配置し、そこからプラントのセンサあるいはアクチュエータへ入出力ケーブルを接続するのが常であった。しかし、この方法ではコントローラの専用設置スペースが必要になることと、コン

トローラの入出力とプラント間の接続ケーブルの敷設に費用がかかるという問題点があった。

【０００３】また、従来のコントローラを分散配置するとコントローラが故障した際にその保守がすぐにはできないという問題点が発生する。

【０００４】また、一般には一つのプラントは複数の制御対象グループから構成され、さらに一つの制御対象グループは複数の制御対象機器から構成されている。そこで、一つのコントローラでどれだけの制御対象機器を掌握するかが従来から議論されている。一般には、一台のコントローラで掌握する制御対象機器が少なければ少ないほど、一台のコントローラが故障してもその影響範囲は小さくなり危険分散化が図れるが、コントローラ間の通信量は増大するという問題があった。

【０００５】一方で、制御の高度化は今後ますます進み、リレー接点ロジックの代替であるシーケンス制御や単なる数値演算制御だけではなく、多変数制御を行うための高速行列演算、あいまい量を取り扱うファジィ制御，学習能力をもつニューラルネットワーク応用制御等各種の制御が複合して用いられるようになってきている。しかし、このような制御システムは、各々専用の機能を持ったコントローラを適当に組み合わせて構築しているのが現状である。その結果、高度な制御を行なおうとすれば、分散システム内に多種類のコントローラを配置する必要があり、設計、プログラミング、運転、保守いずれの時点においても複雑さが増大しそれに伴うコストも増大するという問題があった。

【０００６】また、一つのコントローラを複数のプロセッサで実現する所謂マルチプロセッサ構成のコントローラも実現されている。例えば、伝送処理を行うネットワークコントロールプロセッサ、数値演算プロセッサおよび高速シーケンスプロセッサというように複数のプロセッサでもってコントローラを実現するわけである。しかし、従来の方法では各種制御内容に応じた複数種類のプロセッサを用意し、制御対象に応じてその組み合わせを変える必要があった。即ち、各種制御対象のセンサー、アクチュエータの種類，点数，制御内容が異なるため、各プロセッサおよびプロセス入出力カードがプラグインユニット式で増設、交換できるように作られている。ところが、このコントローラを１ボード化またはＬＳＩ化しようとすると非常に困難な事態が発生する。すなわち、制御対象毎に異る機能のボードやＬＳＩを作ることになり、開発期間、開発コスト面において大きな障害となるとともに保守面においてもすべての品種に対応した保守部品、保守管理が必要になり、保守コストの増大につながる。特にＬＳＩ化のメリットの１つは大量生産による低価格化にあり、この様に個別に多くの種類のＬＳＩを開発することは、このＬＳＩの最大のメリットを享受できないという事になる。

【０００７】また、コントローラのプログラミングに関

しては、従来各制御部のプログラムをユーザが逐一書いていた。一般的には、より良い制御性能を得ようとするとプログラムは複雑化する。また、従来のプログラムでは、経時変化や環境の変化、ユーザの好みに柔軟に対応していくことが大変困難であった。そこで、ニューラルネットなどを使用して徐々に学習しながら制御性を良くするというアプローチがある。しかし、従来の学習によるアプローチでは所望の出力を得るまでに、非常に多くの学習，経験を積まなければならず、学習の初期においては所望とする制御出力とかけ離れたものしか得られないのが常である。一方、制御の初期段階から所望とする制御出力に近い値を得なければならないことも事実である。また、ユーザが持っている知識が優れている場合にはそれを最大限利用するといった方法が望れているが、ニューロ等による学習結果とユーザの知識のどちらが優れているかの判断が難しい。

【０００８】さらに、制御性能の評価については、例えば、自動車の場合のように、その基準は乗り心地，スピード，温度など、複数存在することが多い。また、その価値基準は人によって、また、同じ人でも時によって異なる。このようなことに、柔軟に対応するためには、動作中にユーザがその価値基準算出法を変更可能であることが必要である。

【０００９】ネットワークについては制御対象が定常状態であれば通信情報量の増減はあまり無いが、過渡状態特に異常発生時などは通信情報量が増加してしまう。例えば、従来の分散制御システムの診断方法であるブラックボードアーキテクチャによってもコモンモード故障時に個々のコントローラから通信ネットワーク上に多数の故障情報が送信されネットワークの通信ネックが発生する危険性があった。

【００１０】またさらに、分散制御システムの稼働率と信頼性の向上に関しては、１台のコントローラが故障（ダウン）した場合に他のコントローラがその負荷を代替（バックアップ）する方法が一般的である。例えば、特開昭62－177634号と特開昭63－118860号に示されるように、あるコントローラのダウンに対してバックアップするコントローラをあらかじめ定めておく方法が知られている。さらに、特開平2－224169号と特開平3－296851号に示されるように計算機システムでは各計算機の負荷状況に応じて、ダウンした計算機の負荷を分割し複数の計算機でバックアップする方式が知られている。ところが、前者の方法はバックアップ専用のコントローラを準備する必要があった。また、後者の方法は負荷の分散を集中管理しているためその集中管理用の計算機がダウンした場合、計算機システムの全体が停止する可能性があった。

【００１１】

【発明が解決しようとする課題】本発明が解決しようとする一つの課題は、一台のコントローラで掌握する制御

対象機器を最小限度に少なくし、危険分散を図ることが可能なコントロールシステムを実現することである。

【００１２】またさらに、本発明が解決しようとするもう一つの課題は、コントローラの専用設置スペースおよびコントローラの入出力とプラント間の接続ケーブルの敷設を不要または最小にできる分散設置形のコントロールシステムにも適用できるコントローラを実現することである。

【００１３】またさらに、本発明が解決しようとするもう一つの課題は、できる限り少ない種類で各種の制御対象，制御方式に対応可能な汎用性のあるコントローラを提供することである。

【００１４】またさらに、本発明が解決しようとするもう一つの課題は小型かつ低コストなコントローラを提供することである。

【００１５】またさらに、本発明が解決しようとするもう一つの課題は、小規模のコントローラから大規模のコントローラまで同一アーキテクチャで構成できるスケーラビリティを有したコントローラを提供することにある。

【００１６】またさらに、本発明が解決しようとするもう一つの課題は、ハードウエア完成後に目的に合わせて機能のプログラムが可能なコントローラを提供することにある。ここで言う機能のプログラムとは、数値演算機能、ファジィ推論機能、ニューラルネット処理機能、シーケンス演算機能などの専用処理機能をコントローラ内のハードウエアに埋め込むことである。

【００１７】またさらに、本発明が解決しようとするもう一つの課題は、部分故障に対して自己修復可能なコントローラを提供することにある。

【００１８】またさらに、本発明が解決しようとするもう一つの課題は、プログラミングの簡単なコントロールシステムを提供することにある。

【００１９】またさらに、本発明が解決しようとするもう一つの課題は、コモンモード故障時に個々のコントローラから通信ネットワーク上に送信される故障情報を減じ、ネットワークの通信ネックによるシステムダウンが発生しにくい分散制御システムを提供することにある。

【００２０】またさらに、本発明が解決しようとするもう一つの課題は、集中管理用のコントローラと待機用コントローラを設けずに、正常に動作しているコントローラがダウンしたコントローラの負荷を自律的に分割してバックアップすることが可能な分散制御システムを実現することである。特に、コントローラの多重ダウンに対しても、正常に動作しているコントローラが協調して、その余剰性能を有効に活用し分散制御システムの総合性能の限界までバックアップを実施することにある。

【００２１】

【課題を解決するための手段】上記課題を解決するために、本発明では、コントローラの内部構成、外観構造ともに拡張性のあるものとし、その基本単位をＬＳＩ化し小型のコントローラを実現できるようにした。従って、分散設置型としても集中設置型としても利用できる。また、本コントローラを制御対象機器単位に配置することにより、危険分散システムが実現できる。

【００２２】さらに、汎用性をもったコントローラを実現するために、一つのコントローラを数値演算機能はもとよりニューロ、ファジィ、行列等を高速に演算するための並列演算機能を実行するプロセッサ、シーケンス制御演算を高速に実行するするプロセッサ、さらには、ネットワークとの通信機能を実行するプロセッサ等の複数のプロセッサから構成し、各機能を並列に実行可能とした。また、入出力及びプロセッサを機能プログラムできるチップを実現する方法も提供した。

【００２３】また、同一コントローラ内の一つのプロセッサがダウンしても他のプロセッサがその機能を代替できる手段を設けた。

【００２４】制御演算部にニューラルネットと従来のようなユーザが付与したモデルを計算する手段または方法を設け、両者を重み付けして加えることによって制御出力を生成することとした。そして、その重み付けの値を学習が進んでいない状態ではユーザの付与モデル出力を１００％またはそれに近い値とし、その後学習によって更新し、また、ニューロや他の手段からの出力値に対する重み付けの値もニューラルネットで行い、学習させることとした。

【００２５】また、評価についても、状態の評価値をユーザに表示し、それをユーザが修正する手段を有し、さらに、新しい評価基準をユーザが定義すると共に、複数の評価値を何らかの加工をした後に最小値を求め、これを用いて出力の演算法を更新していくこととした。

【００２６】さらに、故障診断時のネットワークの負荷を低減する手段としては、ネットワークの伝達情報をマンマシン装置に格納し、編集・処理するネットワークモニタ、および各コントローラにつけた入出力情報フィルタを設け冗長な情報はネットワークには乗らないようにした。

【００２７】さらに、ダウンしたコントローラの負荷を自律的に分割してバックアップするための手段は、各コントローラが実行しているタスクとその負荷状況を示す実行分担表と、各タスクを実行したときの負荷を示すタスク負荷表と、各コントローラの負荷状況によりどのコントローラが余裕がありバックアップを実行しやすいかを示す受付順位表とに基づきバックアップを依頼するバックアップ依頼手段と、前記表によりバックアップを受付可能か判定し受付けたか否かを回答するとともに受け付けたタスクの実行を指示するバックアップ受付手段とから構成される。

【００２８】また、一つのコントローラ自身の耐故障性と信頼性を向上するために、各コントローラは複数のプ

ロセッサで構成され、前述の実行分担表はさらに各プロセッサの負荷状況とどのプロセッサが余裕があり負荷をバックアップしやすいかを示すものとし、これらの実行分担表とタスク負荷表に応じて故障したプロセッサの負荷をバックアップするプロセッサを定める内部バックアップ手段を設けたものである。

【００２９】

【作用】一種類で多種多様な制御機能を並列に実行可能なコントローラを実現したことにより柔軟なシステム構成能力を有する分散制御システムが構築できる。即ち、ハード的に同一構造のコントローラを複数台設置することができるため一台のコントローラが故障しても他のコントローラでのバックアップが容易にでき、迅速な保守対応が困難な場所にもコントローラを分散設置できる。

【００３０】プログラミングに関しては、ユーザは付与モデルとして比較的簡単なモデルをプログラムすればよい。学習の初期では付与モデルに従って動作をし、学習が進んでニューラルネットによる学習モデルの出力値が良くなるにつれてニューラルネットの出力に対する重み値を大きくしていくことができる。そしてさらに、状態の評価部を設け、その出力の時間微分によって出力データの評価を行うことで、前記重み値をユーザが意識することなく、学習によって次第に変化させていくことができる。また、該重み値を入力データからニューラルネットで求め、さらにそのニューラルネットを学習させることによって、付与モデルの出力が良いところではその出力を、学習モデルの出力の方が良い場合にはその出力の重みが大きくなるように場合によってその重み付けの仕方を変えることもできるようになる。

【００３１】また、評価値の表示手段と修正手段によって、評価値とユーザの感覚とにずれがあった時に修正することで、簡単にユーザの感覚にあった評価が可能となる。また、ユーザが新しい評価基準を設けることで、さらにユーザの評価方法にできるだけ近付ける。そして、こうしてできた複数の評価基準を直接または何らかの変換をしてその最小値をとることによって、現在ユーザが最も不適と感じている項目を抽出し、この評価基準を用いて制御出力の演算法を更新していくことで、複数の評価基準からみて調和のとれた制御が可能になる。

【００３２】故障診断時のネットワーク負荷を低減する手段に関しては、各コントローラは、故障発生時に異常情報を入出力フィルタに送り、入出力フィルタは、送信された異常情報を含んだ異常情報がすでにネットワークに送信されているかどうかを各コントローラが格納している履歴情報を参照して判定し、送信された故障情報を含んだ異常情報がすでにネットワークに送信されていなければ該異常情報をネットワークに送信することにより、冗長な異常情報がネットワークに送信されることを防止することができる。

【００３３】さらに自律的なバックアップに関しては、各コントローラに具備した実行分担表と，タスク負荷表と，受付順位表とにより、どのコントローラのバックアップ依頼手段でバックアップすべきタスクとバックアップに適するコントローラを検索し、バックアップ受付手段に通知することが可能になる。また、バックアップを依頼されたコントローラのバックアップ受付手段は自己が持つ実行分担表の自己の負荷に基づき、この通知にあるバックアップ対象タスクを実行可能か否かを自ら判断し、選択的にバックアップ対象タスクを受付ける。この選択の結果を受付メッセージとして放送（ブロードキャスト）することにより依頼元および他のコントローラはバックアップを実施したコントローラの負荷状況の変化を実行分担表及び受付順位表に反映可能である。さらに依頼元のコントローラでは受付られなかったバックアップ対象タスクを知ることが可能であり、再び受付順位表（この時は先にバックアップを実施したコントローラは負荷率が高まるため受付順位表の先頭にいないであろう）にしたがって新たなコントローラに未受付分を依頼可能である。

【００３４】このようにして本発明では各コントローラがシステムの実行状況を常に正確に把握し、また、バックアップを依頼されたコントローラが依頼を拒否可能であるため、自律的にバックアップすべきタスクを分割／分散して処理することができる。さらに、一つのコントローラに含まれる複数のプロセッサの一つが故障したときにおいても、内部バックアップ手段が実行分担表により負荷が小さいプロセッサを選択し、そのプロセッサに故障したプロセッサの負荷をすべてバックアップできるかを判定し、そのプロセッサが負荷オーバにならないようにバックアップを実施させ、残りの負荷がある場合にはさらに他のプロセッサを選択し同様にバックアップさせる。このように本内部バックアップ手段は各プロセッサが負荷オーバにならない様に故障したプロセッサの負荷を各プロセッサの負荷限界に収まるように分割し、複数のプロセッサで次々とバックアップさせていくものである。

【００３５】

【実施例】以下本発明の実施例を図面により詳細に説明する。

【００３６】図１は本発明の一実施例であるプラント制御システムを示す。制御対象プラント１００は流量計等のアナログ形センサ１１１，１１５，１１９，制御弁等のアナログ形アクチュエータ１１２，１１６，１２０，リミットスイッチ等のディジタル形センサ１１３，１１７，１２１および電磁弁等のディジタル形アクチュエータ１１４，１１８，１２２から構成されている。本例ではセンサ１１１とアクチュエータ１１２が一組、センサ１１３とアクチュエータ１１４が一組となり各々制御対象機器を構成している。本図では各制御対象機器には一つのセンサと一つのアクチュエータしか図示していない

が、これらは複数個ある場合もある。コントローラ１，２，３，４，５，６は各制御対象機器に対応して配置されそれらはネットワーク１０００で結合されている。

【００３７】マンマシン装置１２００はプラントの運転監視を行い場合によっては人手での運転指令を各コントローラに発する装置である。

【００３８】次に、本発明の一実施例の具体例の一つである鉄鋼圧延制御システムについて説明する。参考のため、従来の鉄鋼圧延制御システムを図２により説明する。ペイオフリール１５１，１５２は被圧延材のコイルであり、被圧延材１８０は入側処理ライン１５３，ルーパ１５４を経由して圧延機１５５，１５６，１５７，１５８にて圧延され最終製品１８１となりテンションリール１５９に巻き取られる。上記リール、圧延機等はモータ１４０，１４１，１４２，１４３，１４４，１４５，１４６で駆動される。各々のモータに対応したドライブ装置１３０，１３１，１３２，１３３，１３４，１３５，１３６は各モータを駆動するパワー回路である。従来はドライブ装置１３０，１３１，１３２，１３３，１３４，１３５，１３６は電力変換器１９０とそれを制御する専用の高速コントローラ１９５から構成されている。図２にはドライブ装置１３０の内部のみ図示したがドライブ装置１３１，１３２，１３３，１３４，１３５，１３６の内部構成も同様である。一方コントローラ１５００，１５０１，１５０２，１５０３は板厚目標値，入側板厚，スタンド間張力などから各モータのスピード指令を計算して各ドライブ装置に指令値を出力する構成を取っていた。

【００３９】本制御システムの運転監視はマンマシン装置１２００，１２０１にて行う。各コントローラ及びマンマシン装置の間はネットワーク１０００で結合されている。

【００４０】図３は本発明を適用した鉄鋼圧延制御システムを示す。本実施例では各モータドライブ装置１６１，１６２，１６３，１６４，１６５，１６６，１６７に一対一でコントローラ１１，１２，１３，１４，１５，１６，１７を割り付けている。しかも、各モータドライブ装置１６１，１６２，１６３，１６４，１６５，１６６，１６７は電力変換器１９０のみから構成されており、従来のような専用のコントローラはない。後述するように本発明のコントローラはマルチプロセッサ構成であり、従来の電力変換器制御機能はそのコントローラ内の一つのプロセッサに割り当てている。

【００４１】なお、図示はしていないが圧延機１５５，１５６，１５７，１５８のロール間ギャップの制御は通常、油圧圧下装置で制御されている。この油圧圧下装置はやはり数ミリ秒という高速で制御しなければならず、従来はそれ専用のコントローラを特別に配置していた。しかし、本発明によれば、モータの制御と同じくその制御に圧下指令計算用コントローラ内の一つのプロセッサを割り当てればよい。このような構成を採ることにより後述するように、非常に高信頼，高性能な制御システムが実現できる。

【００４２】図４及び図５は本発明の他の一実施例であるところの自動車の制御を示したものである。図４は一般的な自動車の制御項目を説明したものである。図５は自動車の制御に本発明を適用した場合の制御システム構成を示したものである。

【００４３】コントローラ２１はエンジン２１ａを制御する。このコントローラ２１はシリンダに流入する空気量と、エンジンの回転数と回転角度と、アクセルペダルが踏み込まれた角度と、エンジンの冷却水の温度と、排気浄化用の触媒の温度とを検出する。これらに応じて、シリンダ内の燃料と空気の比が理論空燃比となるように燃料の噴射量を操作する。そして、上記の状態量に応じて点火時期を操作する。これらの操作はエンジンの効率を高め出力を増加する。同時に、排気中の有害物質を低減する。また、このコントローラ２１はエンジンの冷却水の温度と排気浄化用の触媒の温度が適温より低い場合にはそれらを素早く暖めるために点火時期をやや進角する。コントローラ２１は前照燈の点灯と冷房装置の運転をコントローラ２８がネットワーク１０００を介して通知したとき、エンジンの停止や車速の変動を防ぐためにスロットルをやや開いてエンジンの出力を増加させる。コントローラ２２がネットワーク１０００を介して通知する車速が特定の値を超過した場合には、コントローラ２１は燃料の噴射量を減じて車速が特定の値を大きく越えないようにする。コントローラ２２が変速中であるとネットワーク1000を介して通知したとき、このコントローラ２１は変速の前後で加速度が滑らかに変化するようにエンジンの出力を調整する。コントローラ２１はアクセルペダル踏み込み角度が急激に大きくなった場合には運転車が急激な加速を要求していると判断する。そして、このコントローラ２１は冷房装置で消費されているエンジンの出力を車の加速に使用するためにネットワーク１０００を介してコントローラ２８に対して冷房装置の停止を要求する。同時に、コントローラ２２にシフトダウンを要求する。

【００４４】コントローラ２２は変速機２２ａを制御する。このコントローラ２２は車速を検出し、これに応じたギヤ比に変速する。この時、ネットワーク１０００を介してコントローラ２１が通知するスロットル開度とエンジンの回転速度とコントローラ２３が通知する車体姿勢に応じて、変速の基準となる車速を変更する。スロットル開度が小さくエンジンの回転速度が高いときはエンジンの吸気抵抗による出力の損失が増大するため、低速でシフトアップしエンジン回転速度を低下させる。車体姿勢が後傾しているときは坂道を昇っていると判定しシフトアップの基準車速を高め、低速ギヤを使用するようにして登坂力を維持する。

【００４５】コントローラ２３はサスペンション２３ａを制御する。このコントローラ２３はサスペンションの変位と車体姿勢とを検出し、ネットワーク１０００を介してコントローラ２２から車速を、コントローラ２５から舵角を受信し、これらに応じてサスペンションの空気バネ内の気圧を制御して所望の車体姿勢にする。このコントローラ２３は車体姿勢をネットワーク１０００を介して他のコントローラに通知する。

【００４６】コントローラ２４はブレーキ２４ａを制御する。このコントローラ２４はタイヤの回転速度を検出し、ネットワーク１０００を介してコントローラ２２から車速をコントローラ２５から舵角を受信し、これらに応じて地面に対するタイヤの滑り率を求め、この値が所定の値以下になるように制動力を定める。さらに、旋回時に内輪に弱い制動を掛けて滑らかに旋回する。

【００４７】コントローラ２５はパワーステアリング２５ａを制御する。このコントローラ２５はステアリングシャフトのねじれ角度を検出し、その角度が小さくなるようにパワーステアリングのアシストトルクを調節する。さらに、このコントローラ２５は舵角を検出し、他のコントローラにネットワーク１０００を介して通知する。

【００４８】コントローラ２６は故障を診断する。このコントローラ２６はネットワーク１０００を介して他のコントローラから通知される状態量や操作量を自己のシミュレーション結果と比較することによって、他のコントローラあるいは制御対象機器の故障あるいは劣化を検出する。故障あるいは劣化を検出した場合は、コントローラ２８に対して故障診断結果を通知する。

【００４９】コントローラ２７はドライブレコーダであり、ネットワーク１０００を介して通知された車の動作の履歴を衝撃及び熱に強い記憶装置に格納する。

【００５０】コントローラ２８はスイッチ２８ａとメータ２８ｂとディスプレイ２８ｃなどのマンマシンインタフェースを制御する。コントローラ２８は運転者がスイッチ２８ａを操作した場合にそのオン／オフに応じて対応する装置を作動あるいは停止させる。また、ネットワーク１０００を介して通知された車速やエンジンの回転速度等の状態量と故障診断の結果などをメータ２８ｂあるいはディスプレイ２８ｃに表示する。また、ネットワーク１０００を介した他のコントローラからの依頼あるいは要求に応じて各種装置をオン／オフする。

【００５１】このように各コントローラはネットワーク１０００を介して検出した状態量と制御出力あるいは操作依頼を互いに通知し、そして、各コントローラは通知された状態量と制御出力と操作依頼に応じて所轄の機器を制御する。

【００５２】以上のように複数のコントローラを各制御対象機器毎に対応して分散配置するところに本発明の一つの特徴がある。

【００５３】図６は本発明を階層システムに適用した場合のシステム構成例である。プラントが各々異なった制御対象機器群１００，１０１，１０２から構成されている場合やその制御機器群１００，１０１，１０２が広域に分散している場合などでは本図のようにネットワークをネットワーク１０００，１００１，１００２のように分割し各ネットワーク下にコントローラ１，２，３，４，５，６を配置する。ネットワーク１０００，１００１，１００２は各々統括コントローラ１３００，１３０１，１３０２を介して上位ネットワーク１５００に接続される。統括マンマシン１６００はこの制御システムの運転監視を行う。

【００５４】図７及び図８は今まで述べたコントローラの内部構成を示す。コントローラ１はグローバルメモリ（ＧＭ）５２，演算プロセッサ(ＰＲ１，ＰＲ２，ＰＲ３)５４，５６，５８入出力処理装置（Ｉ／Ｏ）５３，ネットワーク制御プロセッサ（ＮＣＰ）４０，ネットワークコネクタ（NetworkConnector）４１，規模拡張用コネクタ（BusConnector）６０，６１，６２，６３および制御対象機器との接続用コネクタ（Ｉ／Ｏ Connector）６４から構成されている。また演算プロセッサ（ＰＲ１，ＰＲ２，ＰＲ３）５４，５６，５８は各々ローカルメモリ（ＬＭ１，ＬＭ２，ＬＭ３）５５，５７，５９を有している。

【００５５】ＮＣＰ４０はバス７０を介してＧＭ５２，ＬＭ５５，５７，５９およびＩ／Ｏ５３の内容を読み込み他のコントローラまたはマンマシン装置(図１の１２００)へ送信する機能と、逆に他のコントローラまたはマンマシン装置２０００から送信されたデータをＧＭ５２，ＬＭ５５，５７，５９およびＩ／Ｏ５３に書き込む機能を有する。バス７２はプロセッサ５４，５６，５８がＧＭ５２およびＩ／Ｏ５３をアクセスするのに用いる。バス７１は伝送用のシリアルデータ用のバスである。このような構成を採ることでコントローラの拡張性を確保でき、もし一つのプロセッサが故障しても他のプロセッサで代替処理ができ（後述）しかも１台のコントローラ内の全てのプロセッサが故障したとしても他のコントローラで代替処理ができる（後述）システムが構築できる。

【００５６】図８は図７におけるプロセッサ(ＰＲ１)５４の内部構成を示したものである。図７において各プロセッサは同一の構成であり、その構成をプロセッサ(ＰＲ１)を例に取り上げ説明する。プロセッサ（ＰＲ１）５４はタスクを実行する汎用エンジン（ＧＥ）６８１１と、タスクの中に含まれたニューロ演算を加速するためのニューラルエンジン（ＮＥ）６８１２と、タスクの中に含まれたシーケンス制御用の演算を加速するシーケンスエンジン（ＳＥ）６８１３と、アクセスすべきメモリがグローバルメモリ（ＧＭ）５２のときバス７０とプロセッサ内のバス６８１９を結合するバスインタフェース

（ＢＩＦ）６８１７と、ローカルメモリ（ＬＭ）５５とグローバルメモリ（ＧＭ）５２の間で高速な転送を実行するダイレクトメモリアクセス制御（ＤＭＡＣ）６８１８と、スケジューリングのための時計であるフリーランタイマ（ＦＲＴ）６８１５と、各構成要素の故障を検出してプロセッサの動作を停止させるとともに故障を他のプロセッサにバス７２を通して通知する故障制御回路６８２０を含む。さらに、各構成要素はバス６８１９で接続されている。

【００５７】図９は今まで述べたコントローラの他の実施例の内部構成を示す。図７との相異点は図７におけるＮＣＰ４０の機能を複数のプロセッサ８４，８７，９０，９３いずれにおいても実行できるように各プロセッサ８４，８７，９０，９３にシリアルデータのパラレルデータ化あるいはその逆の操作と符号化／復号化などを実行するランアダプタ（ＬＡ）８６，８９，９２，９５を付加したことである。これにより、プロセッサの代替処理と同様に伝送機能（ＮＣＰ）の代替処理が行える。

【００５８】図１０は図９におけるプロセッサ（ＰＲ１）８４の内部構成を示したものである。図８との相異点はランアダプタ（ＬＡ）８６とバス６８１９の接続インタフェース６９０２を設けたことである。

【００５９】図１１はさらに他の実施例の内部構成を示す。図９との相異点は図９におけるローカルメモリ８５，８８，９１，９４の代替としてキャッシュメモリ（ＣＭ）３０１，３０２，３０３，３０４を用いたことである。キャッシュメモリコントローラ（ＣＭＣ）４０１，４０２，４０３，４０４はキャッシュメモリ（ＣＭ）３０１，３０２，３０３，３０４とグローバルメモリ４００との間のデータ転送を行う。

【００６０】図１２は図１１におけるプロセッサ（ＰＲ１）８４の内部構成を示したものである。図１０との相異点は図１２においては図１０におけるダイレクトメモリアクセス制御（ＤＭＡＣ）６８１８が不要なことである。これは、キャッシュメモリ（ＣＭ）３０１とグローバルメモリ４００との間のデータ転送をキャッシュメモリコントローラ（ＣＭＣ）４０１が行うためである。

【００６１】図１３はさらに他の実施例の内部構成を示す。図１１との違いはランアダプタ８６，８９，９２，９５とネットワークコネクタ４１およびプロセッサ８４，８７，９０，９３の接続をプログラマブルスイッチアレイ(ＰＳＷ１，ＰＳＷ２)８１０１，８１０２でプログラマブルにしたことにある。これにより伝送機能をプロセッサ８４，８７，９０，９３いずれにおいても実行できるようだけでなく、シリアルデータのパラレルデータ化あるいはその逆の操作と符号化／復号化などを実行するランアダプタ（ＬＡ）８６，８９，９２，９５の故障に関してもランアダプタを切り替えることによりプロセッサを切り替えることなく伝送機能の代替処理が行える（図１１の構成では、ランアダプタの故障であってもプロセッサを切り替える必要がある。）。尚、図１３におけるプロセッサの内部構成は図１２に示したものと同様で良い。

【００６２】図１４にはプログラマブルスイッチアレイの構成例を示す。スイッチアレイへの入出力ＸｉとＹｊの交点にはＭＯＳトランジスタＴｉｊが設けられており、ＭＯＳトランジスタＴｉｊのゲートには不揮発性メモリセルＭｉｊが接続されている。すなわちメモリセルＭｉｊの出力が“１”の場合ＭＯＳトランジスタＴｉｊがＯＮしＸｉとＹｊが接続され事になる。逆にＭｉｊの出力が“０”の場合はＴｉｊがＯＦＦとなりＸｉとＹｊは切り離された状態となる。またメモリセルＭｉｊにはアドレス線Ａ１，Ａ２，Ａ３とデータ線Ｄ１，Ｄ２，Ｄ３が接続されており、これを介してデータの書込みが行われる。

【００６３】図１５にメモリセルＭｉｊの内部回路例を示す。この回路では不揮発性メモリ素子としてフラッシュメモリを用いている。通常動作状態では書換えモード信号は“ＬＯＷ”になっており、ＮＭＯＳ：Ｎ１がＯＦＦ、ＰＭＯＳ：Ｐ１がＯＮとなっている。またアドレス線の電位は例えば電源電圧に等しくなっている。従ってノードＡの電位はメモリ素子が消去状態（しきい値電圧が低い）のときは“ＬＯＷ”となりインバータＩ２の出力も“０”となる。すなわちこのセルにつながるＭＯＳトランジスタＴｉｊがＯＦＦとなる。逆にメモリ素子が書込み状態（しきい値電圧が高い）のときはノードＡの電位が“ＨＩＧＨ”となりインバータＩ２の出力が“１”スイッチＭＯＳトランジスタＴｉｊがＯＮとなる。メモリセルへの書込みを行うには書込みモード信号を“ＨＩＧＨ”にし、ＮＭＯＳ：Ｎ１をＯＮ，ＰＭＯＳ：Ｐ１をＯＦＦにする。この状態でアドレス線及びデータ線に高電圧を印加することによりメモリ素子Ｍ１は書込み状態になる。メモリ素子の消去はアドレス線を“ＬＯＷ”にしメモリ素子のソースに高電圧を印加することにより行われる。

【００６４】図１６にプログラマブルスイッチアレイに予備領域を設け、部分的な故障に対して救済を可能とするための構成を示す。この例は図１４に示したスイッチアレイに故障救済用素子：Ｔｓ１～Ｔｓ３，Ｔ１ｓ～Ｔ３ｓ，Ｔｙ１～Ｔｙ３，Ｍｓ１～Ｍｓ３，Ｍ１ｓ～Ｍ３ｓを追加したものである。通常故障のない状態ではTs1～Ｔｓ３およびＴ１ｓ～Ｔ３ｓはＯＦＦ、Ｔｙ１～Ｔｙ３はＯＮとなっており、図１４の回路と同じ動作をする。次に故障が検出された場合の救済方法を以下に示す。メモリセルＭ１１が故障した場合について説明する。まずＭ１１～Ｍ１３の内容をＭｓ１～Ｍｓ３に移す。次にＭ１ｓを“１”にし、Ｔ１ｓをＯＮ，TylをＯＦＦにする。これにより故障部分が予備回路で置き換えられたことになり、故障前と同一の動作が可能となる。

【００６５】図１７は図８，図１０，図１２における汎

用エンジンの内部構成例を示したものである。本エンジンはマイクロプログラムメモリ８２０２，読みだし回路8203，マイクロ命令レジスタ８２０４及びデコーダ８２０５、レジスタ、演算ユニット８２０６，マルチプレクサ８２０８，アドレスデコーダ８２０７から構成される。起動当初はアドレスデコーダ８２０７がクリアされ、マイクロプログラムメモリ８２０２のゼロ番地の命令がデコーダ８２０５で解釈され実行される。マイクロ命令には次に実行する命令の番地を指し示している（８２０７）。そのようにしてマイクロプログラムメモリ８２０２に書き込まれているマイクロ命令が順次実行される。

【００６６】マイクロプログラムメモリ８２０２は例えばフラッシュメモリ等の不揮発性メモリで構成されており、このマイクロプログラムを書換えることによりプロセッサの機能を目的にあったものに変更することができる。この書換えのためにマイクロプログラムメモリ８２０２は専用の書換え回路８２０１を内蔵しており、書換えモード信号８３０２により書換えモードに設定することにより書換え用のアドレスバス８３０１，データバス８３００から供給されるアドレス及びデータにしたがって書換えが行われる。

【００６７】図１８に他の実施例として各プロセッサを複数のプロセッサエレメント(ＰＥ)の組み合せで構成した例を示す。各プロセッサの目的に応じて必要な数のプロセッサエレメントを組み合せ各プロセッサを構成する。こうする事によりＬＳＩ全体を同一のプロセッサエレメントで構成することができ、ＬＳＩ化が容易になると共に無駄の少ないコントローラの構成が可能となる。

【００６８】次に入出力部について詳細に説明する。

【００６９】先にも述べたように各コントローラに接続されるセンサ、アクチュエータの種類及び点数は制御対象により異なる。従って従来はそれらに応じた入出力装置をレパートリの中から選択しコントローラに接続する必要があった。一方、本発明では極力少ない種類のハードウエアを用いて制御システムを構築するのが一つの目的であるから以下に示す入出力構成を採用した。

【００７０】図１９は図９におけるＩ／Ｏ８３の内部構成を示す。Ｉ／Ｏ８３はバスインタフェース８３０，Ｄ／Ａ変換器８３２，ＯＰアンプ８３３からなるアナログ出力とＯＰアンプ８３５，Ａ／Ｄ変換器８３４からなるアナログ入力とレジスタ836，パワー回路８３７からなるディジタル出力とパワー回路８３８からなるディジタル入力及びこれらとＩ／Ｏコネクタ６４との結合を決定するパワースイッチアレイ８４０から構成される。

【００７１】パワースイッチアレイ８４０のプログラムはインタフェース回路８３０から信号線８５０を介して行われる。パワースイッチアレイ８４０の構成は図１６で示したものと同一であるが、パワーを取り扱うため物理的な容量は大きくする必要がある。

【００７２】図２０はＩ／Ｏ８３を１チップの半導体で実現した構成図を示す。半導体は大きく２層より構成され上部半導体９８０にはパワー回路部９０２，パワースイッチアレイ部９０１，プロセスとの接続端子９００よりなり下部半導体９５０はＯＰアンプ部９５１，Ｄ／Ａ変換部９５２，Ａ／Ｄ変換部９５３，レジスタ部954，バスインタフェース９５５及びバスインタフェースとの接続端子９６０から構成されている（図２１は下部半導体９５０を図示したものである）。また、上部半導体９８０と下部半導体９５０の配線は配線層９７０によって行う。

【００７３】図２２及び図２３は図９におけるコントローラ１′を１チップの半導体で実現した構成図を示す。半導体は大きく２層より構成され上部半導体９８０は図２０と同じ構成であるが、下部半導体９５０はＯＰアンプ部９５１，Ｄ／Ａ変換部９５２，Ａ／Ｄ変換部９５３，レジスタ部９５４以外にメモリ群９５６およびプロセッサ群９５７から構成されている（図２３は下部半導体９５０を図示したものである）。また、上部半導体９８０と下部半導体９５０の配線は配線層９７０によって行う。

【００７４】以上コントローラの内部構成について実施例を用いて説明したが、本発明の範囲は実施例の範囲に限定されるものではなく種々の変形が可能である。例えば機能プログラム用の不揮発性メモリとしては、フラッシュメモリ以外に他の電気的に書換え可能なＥＥＰＲＯＭまたは強誘電体ＲＡＭ、さらには電池でバックアップしたＳＲＡＭでもよい。

【００７５】次に図２４に本発明のコントローラの外観を示す。コントローラの前面にはＩ／Ｏコネクタ６４が左右側面と上下面には規模拡張用コネクタ６０，６１，６２，６３が実装されている。背面１７００にはネットワーク用コネクタ１７０１及び電源用コネクタ１７０２が実装されている。外形寸法は高さ約２０cm，奥行き約１０cm，幅約３cmあり、入出力を接続可能でかつ分散設置するに十分な寸法である。本寸法は本発明によりどの程度の小型化を図れるかを参考に示すためのものであり、本発明はこの寸法に限られる訳ではない。

【００７６】図２５は二台のコントローラを規模拡張用コネクタ６０，６１を用いて結合したものである。

【００７７】図２６は二台のコントローラを規模拡張用コネクタ６２，６３を用いて結合したものである。

【００７８】図２７は四台のコントローラを規模拡張用コネクタ６０，６１，６２，６３を用いて結合したものである。

【００７９】このように本実施例のバス構成と外観構造を有することによりコントローラ規模が容易に拡張できる。

【００８０】次に、本発明で用いた制御演算方法について説明する。

【００８１】図２８はセンサー等からの入力データに基づいて制御演算を行う制御演算部の基本構成図を示す。本発明では、入力データ２００１を複数の出力生成手段、ここではニューラルネットワーク２１０１とファジィ推論手段２１０２とＰＩＤ制御手段２１０３の３個の手段に入力する。そして、重み付け部２０２１において各出力生成手段の出力値をそれぞれに対応する重み値K_1，K_2，K_3で重み付けし、出力データ合成部２０２２で重み付けした各出力値を加算し、出力データ２００２を得る。この時、前記重み値K_1，K_2，K_3を加えて１になるようにしておけば、出力データ合成部２０２２では、重み付けされた出力を単純に加算するだけでよい。ここで、ニューラルネットワーク２１０１，ファジィ推論手段２１０２，ＰＩＤ制御手段２１０３の３個の手段は各々並列的に実行できるため各々別プロセッサで実行させる（例えば、図７のコントローラでいえば、プロセッサ５４，５６，５８に３種の演算を担当させる）ことにより、処理速度を落とすことなく制御演算が実行できる。

【００８２】図２９に実施例の全体概要図を示す。ここでは、出力計算部２０１２及び出力評価部２０１５よりなり、出力計算部２０１２および評価部２０１５がそれぞれ学習モデルと付与モデルを持って両者を適応的に協調させる制御方法を示す。センサ等２０１０からのデータ２００１を必要に応じて入力処理部２０１１で入力処理した入力データ２３００は出力計算部２０１２と評価部２０１５へ入力される。出力計算部２０１２では、入力データ２３００を受けとり、その内部に持つパラメータ等を用いてその出力データ２３０１を計算し、必要に応じて出力処理部２０１３にて必要な出力処理を行った後、その値２００２をアクチュエータ等２０１４へ出力する。ここで、パラメータ等とは、学習モデルであるニューラルネット内のニューロン間の結合の重み値や、付与モデル内のパラメータ、さらに学習モデルと付与モデルの重み付けのパラメータを含む。一方、評価部２０１５では、やはり、入力データ２３００及び内部のパラメータ等を用いて現在の状態の評価を行い、その結果から出力評価信号２３０２を出力する。そして、出力計算部２０１２ではこの出力評価信号２３０２を用いて、この値が大きくなるように内部のパラメータ等の学習を行う。一方、評価部２０１５内のパラメータに関しては、例えばマンマシン装置１２００に状態評価信号２３０３を表示し、ユーザがこれを見て不適切と感じる場合にタッチパネル、マウス等のポインティングデバイス２０１７を用いて修正値を入力する。これを外部評価信号２３０４として評価部２０１５へ入力し、この時に外部評価有効信号２３０５をオンにして評価部２０１５内のパラメータ等を更新することによって、様々な状況に適応した評価ができるように学習させていく。

【００８３】出力計算部２０１２の構成を図３０に示す。出力計算部２０１２の学習は、評価部２０１５の場合の外部評価信号２３０４のような直接的な教師信号はないため、各パラメータ等に乱数成分を付加し、その時の評価値に基づいて乱数成分を評価し、修正を行う。従って、出力の計算法は以下のようになる。まず、入力データ２３００と学習モデルパラメータ保持部２４２１から得られる学習モデルパラメータを用いて、学習モデル計算部２４０１において出力を計算すると共に、予めユーザから付与された付与モデルに従って、入力データ２３００と付与モデルパラメータ保持部２４１１から得られる付与モデルパラメータに乱数付加部２４１２によって微小な乱数を付加したものをパラメータとして用い、付与モデル計算部２４０２の出力を求める。そして、重み付け部２４０３でそれぞれの出力値を重み付けして加えあわせて出力データ２３０１とする。ただし、学習モデル計算部２４０１からの出力には乱数付加部２４２２によって微小な乱数を加えたものに対し重み付けを行う。また、さらに、重み付け部２４０３で用いる重み値も、重み値保持部２４３１に保持されている重み値に対し、乱数付加部2432によって乱数を付加したものを実際に用いて重み付けする。まず、重み付け部2403の重み値に関しては、乱数付加部２４３２によって付加された乱数２４３３と評価部２０１５から得られる出力評価信号２３０２を重み値更新量計算部２４３４において、例えば乗算し、重み値保持部２４３１にその値を渡して、もとの重み値に加えて新しい重み値とする。一方、学習モデル計算部２４０１内のパラメータに関しては、出力誤差計算部２４２４にて、乱数付加部２４２２によって付加された乱数２４２３と評価部２０１５からの出力評価信号２３０２を例えば乗算して、学習モデル計算部２４０１に与える誤差とし、学習モデル計算部２４０１では、その値に基づいてバックプロパゲーション等の学習アルゴリズムに従って内部のニューラルネットの重み値等のパラメータに更新量を計算し、新たな値を学習モデルパラメータ保持部２４２１に書き込む。また、付与モデル計算部2402では、やはり、付与モデル内パラメータ更新量計算部２４１４において、乱数付加部２４１２で加えた乱数２４１３と評価部２０１５からの出力評価信号2302を例えば乗算し、その値を付与モデルパラメータ保持部２４１１に渡して新しい付与モデルパラメータの値を書き込む。

【００８４】一方、図３１に評価部２０１５の構成を示す。ここでは、まず、入力データ２３００と学習モデルパラメータ保持部２５２１から得られるニューラルネットの結合の重み値などの学習モデルパラメータを用いて、学習モデル計算部2501から出力を求める。また、それと並行して、予めユーザから付与されたモデルに従って、入力データ２３００と付与モデルパラメータ保持部２５１１から得られる付与モデルパラメータを用いて、付与モデル計算部２５０２においても出力を計算する。

そして、重み付け部２５０３でそれぞれの出力値を重み値保持部2531に保持されている重み値に基づいて重み付けして足しあわせ、その足しあわせた結果を時間微分計算部２５０５にて時間微分することによって出力評価信号2302を生成する。また、評価部２０１５内の学習に関しては、外部評価有効信号2305がオンの時に外部評価信号２３０４からの信号に基づいて行われる。例えば、学習モデル計算部２５０１と付与モデル計算部２５０２のそれぞれの出力を比較し、外部評価信号２３０４に近い方の重み付けパラメータを大きくし、もう一方のパラメータを小さくする。また、学習モデル計算部２５０１に対しては、外部評価信号２３０４を教師信号として与えて学習を行う。一方、付与モデル計算部2502では、通常の出力値以外に、パラメータに微小な乱数を与えてもう一つの出力値を求め、その両者を比較して乱数を加えた方が外部評価信号２３０４に近ければ、実際にパラメータに対し、加えた乱数と学習定数を掛けたものを足し、乱数を加えない方が良い場合には、パラメータから加えた乱数と学習定数を加えたものを引くことによって学習を行うことができる。また、付与モデルは学習機能がなくても良いし、評価部２０１５が付与モデル計算部だけで学習モデル計算部がなくても差し支えない。また、学習モデル計算部２５０１と付与モデル計算部2502の両方を有する場合には、初期状態の両者の出力の重み付けは、付与モデル側を１とし学習モデル側を０とすることによって、学習開始前には学習モデルの出力が影響を及ぼさないようにすることができる。

【００８５】次に、複数個の評価基準を有する場合の実施例を示す。全体構成図を図３２に示す。ここでは、複数の評価部として第１評価部２６０１と第２評価部２６０２があるものとする。各評価部内のパラメータの修正は、評価部が１個の場合と同じく、マンマシン装置１２００に第１評価部２６０１の状態評価値２６０３と第２評価部２６０２の状態評価値２６０４を表示し、これをタッチパネル、マウス等のポインティングデバイス２０１７を用いて修正し、修正された時は、該当する外部評価有効信号２６０５に１を立てる。そして、その時の修正値を外部評価信号２６０６として各評価部へ受け渡す。そして、該当する評価部では、その内部のパラメータを修正すれば良い。また、出力評価信号に関しても複数得られるが、例えば、その複数の出力評価信号の最小値を選んで出力計算部２０１２へ与えることが考えられる。すると、ユーザが最も不適であると感じる評価基準に関して、より良くなるように出力計算部２０１２内のパラメータ等を更新できることになる。

【００８６】次に、複数の手段や方法による出力の重み付けをする重み値を、入力データ２００１によって変化させる場合を図３３に示す。ここでは、入力データ2001をニューラルネットワーク２７０１に入力し、出てきた出力に乱数付加部２７０２によって乱数成分を付加し、その値Ｋに基づいて重み付け部２０２１において、ニューラルネットワーク２１０１と出力生成手段２１０２の出力を重み付けし出力データ合成部２０２２で重み付けされたデータを加えあわせて出力データ2002を生成する。一方、教師信号生成部２７０３においてその乱数成分と出力評価信号２３０２を掛け算しニューラルネットワーク２７０１の出力と足して教師信号としてニューラルネットワーク２７０１に渡し、学習を行わせる。ただし、この時、学習が始まる前はニューラルネットワーク２７０１の出力が０に近い値になるようにし、出力生成手段２（２１０２）の出力が出力データ２００２になるようにすると良い。

【００８７】以下、本発明のネットワーク通信負荷を低減する方法に関する実施例を図面を用いて詳しく説明する。図３４は、本発明の対象となる分散制御システムの構成を示したもので、ネットワーク１０００によって複数のコントローラ（１，２，３，…）が制御を行っている。システム全体の制御情報および診断情報は、マンマシン装置１２００において加工・表示される。各コントローラは、それぞれに制御を実行するだけでなく、診断処理も実行するので、診断処理部４０１０，制御実行部４０１１，制御情報および診断情報をネットワークを通じて他のコントローラに送信するための通信インタフェース４０１２をもっている。

【００８８】このような分散制御システムに於ては、コモンモード故障が発生した場合に、各分散制御センサーからの異常情報が多数ネットワークに送信され、通信不良となる可能性がある。

【００８９】例えば、各コントローラが制約条件をもっており、診断処理部において制約のチェックをおこない、もし、制約が充足されなければ制約を充足しない制約式をネットワークに送信する場合を考える。このような場合、異常診断とは充足されない制約式の制約変数の中でどの変数が制約を充足しない原因となっているかを同定することである。

【００９０】図３５は、以後の説明のための具体例として４つのコントローラからなる分散制御システムの各コントローラの入力変数、出力変数、制約式を示したものである。例えば、コントローラ１は、入力変数としてＸ１，Ｘ２をもち、出力変数としてＸ３をもち、制約式として、Ｃ１：Ｅ（Ｘ３，Ｘ１＋Ｘ２，ｅ１）をもつとする。ここで、Ｅは、誤差関係式で、第１変数と、第２変数の絶対値の差が第３変数以下であることを表す。この例では、

Ｘ１＋Ｘ２－ｅ１＜Ｘ３＜Ｘ１＋Ｘ２＋ｅ１

であることを示している。

【００９１】対象プラントの状態量がＸ１＝１，Ｘ２＝２，Ｘ４＝２とする。コントローラが、この各状態量に対して制御を行う時、図３５に示した制約をおのおのがチェックして、もし制約が充足されない場合には、ネッ

トワークに情報を送信するとしよう。この時、コントローラ１が正常に動作していれば、入力変数Ｘ１，Ｘ２の値から、制約条件Ｃ１を充足するＸ３の値であるＸ３＝３を決定し、出力する。しかし、ここで、コントローラ１が何等かの原因で故障し、Ｘ３＝１３を出力したとしよう。この時、コントローラ２では、制約条件Ｃ２と、入力Ｘ３，Ｘ４から、Ｘ５＝１５，制約条件Ｃ３と入力Ｘ３から、Ｘ６＝１３が出力されるが、制約条件Ｃ４が充足されないので制約条件Ｃ４が未充足であるという異常が送信される。コントローラ３では、入力Ｘ３，Ｘ５と、制約条件Ｃ５より、出力Ｘ７＝１９５となるが、これは、制約条件Ｃ６を充足しないのでコントローラ３からは、制約条件Ｃ６が未充足という異常が送信される。コントローラ４では、入力変数Ｘ３，Ｘ５およびＸ６と制約条件Ｃ７よりＸ９＝１５が出力され、制約条件Ｃ８より、Ｘ８＝２６が出力されるが、これは、制約条件Ｃ９を充足しないので、コントローラ４からは、制約条件Ｃ９が未充足という異常情報が送信される。このように、分散制御系においては一つのコントローラが故障したために、他の多くのコントローラから異常情報が出力され通信ネックが発生し、診断が妨げられる心配がある。

【００９２】このような問題を解決するために、本発明においては、個々のコントローラからネットワークへ入出力される情報をフィルタリングする方法をとった。また、異常発生時に送信する情報として、各コントローラがフィルタリングを実行するためにマンマシン装置に送るものと、ネットワークを経由して順次各コントローラに送るものの二つデータフレームを用意する。ネットワークもこれにあわせて、各コントローラをリング状に結合するものと、各コントローラとマンマシン装置をつなぐ二種類のネットワークを用意すれば、より高速な処理が可能である。ここでは、一実施例として、マンマシン装置も各コントローラも一つのネットワークで接続しており、異常発生時の異常情報の送り先を異常情報の中に含める方法をとる。

【００９３】図３６は、個々のコントローラの通信インタフェースの構成を示したもので、４０３０が不要な入力情報をふるい分け、必要な情報だけコントローラに取り込む為の入力情報フィルタ部、４０３１は、不要な出力情報をふるい分け、不要な出力情報をネットワークに送信しないための出力情報フィルタ、４０３２は、暗号化され、情報圧縮された情報を復元するための通信処理部である。これらの処理は図７で示すコントローラではＮＣＰ４０が、図９，図１１に示すコントローラではランアダプタ８６，８９，９２，９５が実行する。

【００９４】ここで、不要な出力情報のふるい分けの一実施例として、制約変数集合を用いた方法を説明する。ここで、制約変数集合とは、制約式中に定義された全ての変数をさす。例えば、図３５中の制約条件Ｃ１の制約変数集合をＶ（Ｃ１）とすると、Ｖ（Ｃ１）＝｛Ｘ１，Ｘ２，Ｘ３｝である。誤差変数ｅ１は誤差範囲を示すために人工的に導入されたものなので、制約変数集合には含めないとする。

【００９５】この制約変数集合の包含関係に基づいて、個々の分散制御系で検出した異常をネットワークを経由して、他の個別コントローラとマンマシン装置１２００に送信するか否かを決定する。ただし、この情報だけからでは不十分なので、個々の分散制御システムにおける入出力変数の情報も利用する。つまり、ネットワークに異常情報を送信するときは、図３７に示したように、制約変数集合の他に、入力変数集合、展開変数集合、消去変数集合，異常情報データＩＤ，機器ＩＤの情報も付加する。ここで、入力変数とは、異常が発生した個々のコントローラにおける入力変数で、コントローラ１における入力変数を、Ｉｎ（１）とすると、Ｉｎ（１）＝｛Ｘ１，Ｘ２｝である。展開変数と、消去変数は、別のコントローラから異常情報を受信した場合に異常変数の絞り込みの際につけ加わる情報で、後で詳しく述べる。異常データＩＤは、このデータフレームが、異常データというタイプのデータで、いつ送信されたかを示す為の情報で、ここでは、ＡＢ＿９２＿０７＿０３＿１０＿２４＿１０とする。ＡＢは、異常情報であることを示し、後の数字は異常情報を送信したのが、９２年７月３日１０時２４分１０秒であることを示す。送信先ＩＤは、異常データが、マンマシン装置あってのものか、各コントローラ宛のものかを示すものである。例えば、マンマシン装置向けのものであれば、ここに、０を、各コントローラ宛のものであれば、ここに、また、機器ＩＤは、この異常情報が発せられたコントローラのＩＤを記述するためのデータフィールドでここでは、コントローラ１から送信されたので１と記入されている。

【００９６】このような異常情報がネットワークに送信されると、各コントローラは、その情報が自分と関係のあるものかどうかを、入力情報フィルタにおいて判定し、もし関係のある情報であれば取り込む。一方、異常情報データフォーマット中の異常データＩＤと制約変数集合、機器ＩＤ，入力変数集合だけは、加工されないまま、ただちに、マンマシン装置に送られる。

【００９７】以下の説明では、今の例をもちいて、まず、どのようにして不必要な異常情報が、ネットワークに送信されるのが抑制されるかを示し、次に、異常箇所がどのように同定されるかを示す。

【００９８】まず、不必要な情報の抑制処理を図３８及び図３９に示す。図３８は、処理の概要を示したもので、マンマシン装置に登録されている異常情報を参照して、新たに送信しようとしている異常情報が既に送信されている情報に含まれているかどうかを判定し、もし含まれていれば、送信を止める。図３９は、異常情報相互の包含関係の判定処理を示したもので、制約集合の間に

包含関係があるかないかをまず調べ、もし包含関係がなくてもコントローラ間に入出力関係があり、この入出力関係まで考慮すると包含関係が成り立つ場合もあり、この判定も行っている。これらの処理を具体的な例で示す。

【００９９】まず、コントローラ１から上記の異常情報がマンマシン装置宛のものと各コントローラ宛のものの２つのデータフレームとして送信され、すでにマンマシン装置で受信している場合について考えてみる。すでに述べたように他のコントローラ２，３，４でも制約違反を検知し異常情報を送信しようとする。たとえば、コントローラ２では、制約条件Ｃ４が充足されない。この異常情報の制約変数集合はＶ（Ｃ４）＝｛Ｘ５｝であり、入力変数集合は、｛Ｘ３，Ｘ４｝である。コントローラ２の通信インタフェースの出力情報フィルタ部では、この情報を送信する前に、マンマシン装置の異常情報を参照して、自分が発信しようとしている異常情報の制約変数集合を含む制約変数集合をもった異常情報がすでに送信されていないかどうか確認する。この場合、コントローラ１から送信された異常情報の制約変数集合のＶ（Ｃ１）＝｛Ｘ１，Ｘ２，Ｘ３｝は、Ｖ（Ｃ４）を含んでいない。しかし、入出力関係まで考慮すると異なった結果となる。つまり、コントローラ２の入力変数Ｘ３は、コントローラ１から送信された異常情報の制約変数集合Ｖ（Ｃ１）に属しているから、この入力変数Ｘ３の影響でコントローラ２の出力変数Ｘ５，Ｘ６に異常が生じる可能性がある。

【０１００】一方、コントローラ１の異常情報の入力変数は、｛Ｘ１，Ｘ２｝であるから、コントローラ２の出力変数Ｘ５に関する異常の原因は、コントローラ１の出力変数Ｘ３を媒介してコントローラ１の入力変数Ｘ１，Ｘ２である可能性がある。つまり、コントローラ２にとっては、Ｖ（Ｃ１）＝｛Ｘ１，Ｘ２，Ｘ３｝にくわえて、Ｏｐ（Ｃ１，２）＝｛Ｘ５，Ｘ６｝も制約変数である。ここで、Ｏｐ（Ｃ１，２）は、コントローラ２にとっての制約条件Ｃ１の仮想的な制約変数であって、ここでは展開変数集合と呼ぶ。そこでＶ（Ｃ１）U_{op}（Ｃ１，２)にＶ（Ｃ４）が含まれているから、制約条件Ｃ４の未充足は、すでに送信された制約条件Ｃ１の未充足に関する情報に含まれる情報であって、ネットワーク上にデータがあふれることを防止するためには、出力情報フィルタ部でふるいにかけ、送信させないこととする。

【０１０１】同様にコントローラ３においても制約条件Ｃ６が充足されていないのでこれをネットワークに送信しようとする。ところがすでに、コントローラ１から異常情報がマンマシン装置に送信されているので、コントローラ３の出力情報フィルタでは、自分の持っている異常情報が送信するに値するものかどうか判定する。この時、コントローラ２におけるのと同様に、Ｏｐ（Ｃ１，３）を求めると、コントローラ３は、入力変数としてコントローラ１の出力変数Ｘ３を持っているからコントローラ３の出力変数Ｘ７も仮想的な制約変数であって、Ｏｐ（Ｃ１，３）＝｛Ｘ７｝となり、Ｖ（Ｃ１）U_{op}（Ｃ１，３）にＶ（Ｃ６）が含まれるから、制約条件Ｃ６に関する異常情報は、送信されない。さらに、コントローラ４についてもＯｐ（Ｃ１，４）＝｛Ｘ８，Ｘ９｝であり、Ｖ（Ｃ１）U_{op}（Ｃ１，４）にＶ（Ｃ９）が含まれる。したがって制約条件Ｃ９が充足されないための異常情報は送信されない。このようにして、重複した異常情報の発生を押えることができる。

【０１０２】次に、既に送信された異常情報を用いて、異常情報を絞り込んでいく方法について図４０によって説明する。異常情報が、あるコントローラから送信されるとデータは、直接、マンマシン装置に行くものと、各コントローラに放送されるものに分かれる。異常情報の絞り込みが行われるのは、各コントローラに放送される異常情報である。まず、各コントローラはネットワーク上の情報が、自分と関係のある情報かどうかの判定をおこなう。

【０１０３】まずデータバス上の異常情報をコントローラの通信インタフェースに取り込み、取り込まれた異常情報は、ネットワークから削除する。通信インタフェースでは、まず通信処理部において、データを圧縮された状態から処理可能な状態に変換し、次に異常情報データの中の入力変数を出力変数として持つかどうかを判定し、もし持てば、コントローラの入力変数を展開変数として異常情報に付加する。今の例では、コントローラ１から発生した異常情報の入力変数集合は、Ｘ１，Ｘ２であり、これらをコントローラ２は、出力変数としてもたないからＮｏである。次に、異常情報の制約集合に含まれる制約変数集合に含まれる制約変数集合にもつ制約条件はあるかを判定し、これもＮｏであるから、異常情報を加工することなく、ネットワークに異常情報が返される。つまり、入力フィルータでフィルタされ、何も処理がおこなわれない。今の例では、他のコントローラも同様にして、何の処理も行わず、始めに送信された制約Ｃ１の異常情報が原形のまま、ネットワークをとおりぬける。一方、既に述べたようにこの場合には、ほかの制約違反は、出力情報のフィルターにかかって、発信されないから、絞り込みがなくても、はじめから原因は、コントローラ１にあると特定することができる。これに対して、何かの原因で、例えば、コントローラ３のＣ６の制約条件違反の異常情報が、コントローラ１からよりも先に送信されたとする。この異常情報の制約変数集合は、Ｖ（Ｃ６）＝｛Ｘ７｝であり、入力変数集合Ｉｎ（３）＝｛Ｘ３，Ｘ５｝である。この異常情報が、コントローラ１にくると、入力変数Ｘ３をコントローラ１は、出力変数としてもつから（このような変数を共通変数という）、コントローラの入力変数Ｘ１，Ｘ２，およびを展開変数に追加する。次の判定で、異常情報の制約

変数集合｛Ｘ７｝と展開変数集合｛Ｘ１，Ｘ２，Ｘ３｝の和集合は、制約条件１の制約変数集合｛Ｘ１，Ｘ２，Ｘ３｝を含むから、異常情報は、図４１の処理手順によって加工される。このとき、制約条件Ｃ１は、制約条件Ｃ６に含まれるという。まず、受信した制約条件に含まれる制約条件Ｃ１が充足しているかどうかを判定する。この場合は、充足していないので、次に、制約条件Ｃ１は、送信されているかどうかをチェックする。Ｃ１が先に送信された場合には、Ｃ６は、すでに述べたプロセスにより送信されないが、Ｃ６が先に送信された場合には、Ｃ１が送信されることはありえる。Ｃ１がすでに送信されていれば、受け取った異常情報は、消去され、まだ送信されていなければ、異常情報は、Ｃ１に関するものに書き換えられる。いずれの場合にもネットワーク上には、Ｃ１に関する制約条件のみが生き残る。これは、見方をかえれば、異常原因がコントローラ１に絞り込まれたことをしめしている。

【０１０４】なお、以上のような不必要な情報の送信のカットと、異常原因の絞り込みは、各コントローラの入出力フィルタの部分及び、マンマシン装置に異常情報の包含関係に関する知識をインストールすることによっても実現できる。

【０１０５】例えば、制約の包含関係を図４２に示したようなツリーで表現した知識を用意し、図３７の異常データフォーマット上の制約変数集合のかわりに、充足されなかった制約名称を記述することとすれば、図３９の処理の代りに既にマンマシン装置に登録されている異常情報の制約式名称を用いて、既に登録されている異常情報が送信しようとしている異常情報を含んでいれば、異常情報を他のコントローラに送信せずに、統合監視装置の異常情報を追加するだけにすることにより、不要な異常情報の送信を抑制できる。

【０１０６】さらにまた、信頼性及び耐故障性の向上を目的とした本発明の分散制御システムのもう一つの実施例を示す。本発明は分散制御システムのコントローラの多重ダウンに対しても稼働中のコントローラが負荷を自律的に分散しバックアップすることを可能にしたものである。このバックアップを実現する手段を図４３に示す。図４３は分散制御システムのコントローラの多重ダウンに対して稼働中のコントローラが負荷を自律的に分散しバックアップする本発明の一実施例である。プラント１００のセンサ及び制御対象に接続されそれらを制御するコントローラ１，２，３，４はネットワーク１０００で相互に接続された分散制御システムを構成している。コントローラ１，２，３，４はそれぞれ通信手段６１１０，６２１０，６３１０，６４１０と、バックアップ依頼手段６１２０，６２２０，６３２０，６４２０と、バックアップ受付手段６１３０，６２３０，６３３０，６４３０と、スケジューラ６１４０，６２４０，６３４０，６４４０と、制御手段６１５０，６２５０，６３５０，６４５０と、記憶手段６１６０，６２６０，６３６０，６４６０と、さらに、内部バックアップ手段６１７０，６２７０，６３７０，６４７０と、切り離し手段６１８０，６２８０，６３８０，６４８０とを含む各コントローラは同様の構成であるので、コントローラ１を代表例として取り上げ説明する。なお、各コントローラは後述の図４７に示すとおり４個のプロセッサを持っている。さらに各プロセッサはここに示した通信手段、バックアップ依頼手段、バックアップ受付手段、内部バックアップ手段、切り離し手段、スケジューラ、制御手段の各手段を提供するに充分なハードウェアを持ち、かつ、各手段を提供するソフトウェア（タスク）を実行するに充分な処理能力を持っている。このため前記の手段は何れのプロセッサ上でも実施可能である。以下各手段の機能を説明する。

【０１０７】通信手段６１１０はネットワーク１０００、バックアップ依頼手段６１２０，バックアップ受付手段６１３０，スケジューラ６１４０，制御手段６１５０と記憶手段６１６０，内部バックアップ手段６１７０，切り離し手段６１８０に接続されており、他のコントローラとのメッセージの交換を実行し、コントローラ内の他の手段と他のコントローラの間でのメッセージあるいはデータの交換を可能にする。スケジューラ６１４０は自分自身に登録された複数のタスクをプロセッサに割り付ける順番及び時間間隔などを定め、かつ指示するものである。制御手段６１５０はプラント１００のセンサ及び制御対象を制御するものである。スケジューラ６１４０と制御手段６１５０は記憶手段６１６０と接続されており、そこに記憶されたタスク及びデータとスケジューリング情報に基づきスケジューリングとプランと１００の制御を司る。また、スケジューラ６１４０と制御手段６１５０は通信手段６１１０と接続されており、他のコントローラとの同期をとるために通信手段６１１０とネットワーク１０００を介して他のコントローラとメッセージを交換する。本実施例は以上により分散制御を実施している。

【０１０８】バックアップを司る手段について説明する。

【０１０９】切り離し手段６１８０は内部バックアップ手段６１７０と通信手段６１１０に接続されている。この手段はコントローラ内部のハードウェアの故障あるいはソフトウェアの暴走などのダウンを検出し、ダウンしたプロセッサの動作を停止させる。また、ネットワーク１０００を通して他のコントローラから自己を含むコントローラの故障を通知された場合にも同様の動作を実施する。他のコントローラからの故障の通知によってバックアップを開始できるように、切り離し手段６１８０は通信手段６１１０に接続されている。さらに、自ら故障を検出した場合および故障を通知された場合のどちらでも、どのプロセッサがダウンしたかを内部バックアップ

手段６１７０に前記の接続を通して知らせる。

【０１１０】前述のスケジューラ６１４０は、前述の機能に加えて、コントローラ内部のプロセッサで実行されている各タスクに対して単位時間当りの実行時間を測定し、その値の単位時間に対する比を各タスクの負荷率とする。この値により後述するタスク負荷表を更新する。そして、これらの負荷率の和が後述する限界負荷率を越えた場合に当該のプロセッサにて負荷オーバが発生すると予測する。また当該のプロセッサにて各タスクの負荷率の和が１００％を越えたか否かを判定することにより負荷オーバを検出する。越えた場合が負荷オーバである。負荷オーバが発生すると予測されたプロセッサあるいは負荷オーバが発生したプロセッサを内部バックアップ手段６１７０に知らせる。

【０１１１】内部バックアップ手段６１７０は通信手段６１１０，記憶手段６１６０，切り離し手段６１８０，スケジューラ６１４０とバックアップ依頼手段６１２０に接続されている。この手段は、切り離し手段６１８０からコントローラ内部のダウンを通知された場合に、あるいはスケジューラ６１４０からコントローラ内部の負荷オーバの発生あるいは負荷オーバの予測を通知された場合に図４４の手順に従い、ダウンあるいは負荷オーバにより実行できなくなるタスクをその優先度に応じてコントローラ内でバックアップする。部分的ダウンあるいは負荷オーバがコントローラ全体のダウンにつながることを防ぐために、コントローラの管理運用に必要不可欠なタスクを最も優先する。ネットワーク１０００の通信量のボトルネックが顕在化し通信の遅延あるいは不良が頻発することを防ぐために、他のコントローラでバックアップした場合に多量の通信を必要とするタスクが次に優先される。バックアップは負荷の軽いプロセッサから順番に実施していき、バックアップされるべきタスクがあるかぎり、次々と負荷の重いプロセッサで実施していく。途中でバックアップされるべきタスクが無くなればそこで処理を終了する。すべてのプロセッサでバックアップを実施してもバックアップされるべきタスクが残っているときはバックアップ依頼手段６１２０にこの状況を通知して図４５のコントローラ間でのバックアップに関する手順を実行させ、本手段は実行を終了する。この図４４の手順を内部バックアップ手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【０１１２】まず、本手段は手順６５１０から手順６５１５にて、ダウンあるいは負荷オーバにより実行できなくなったタスクのバックアップを担当するプロセッサをコントローラ内から決める。この決定において記憶手段６１６０に格納されている各プロセッサの実行状態に関する情報を用いる。すなわち、多くのタスクをバックアップさせるために負荷の低いプロセッサあるいは余剰性能の大きいプロセッサに決定する。なお、これらの情報は図４８に示した実行分担表と呼ぶ表に格納されている。さらに、ダウンあるいは負荷オーバにより実行できなくなったタスクのうちバックアップされるべきタスクを決定する。この決定では、タスクの属性に関する情報を用い、他のプロセッサでバックアップする必要のないタスクを除外する。このタスクの属性に関する情報は図４９に示したタスク負荷表と呼ぶ表に格納されている。

【０１１３】つづいて本手段は手順６５２０から手順６５４０にて、バックアップされるべきタスクにバックアップを担当する部分がバックアップ前に実行していた複数のタスクを追加する。そしてこれらの中から、バックアップを担当するプロセッサがバックアップ後に実行すべきタスクを優先度に応じて選択する。このとき、部分的ダウンあるいは負荷オーバがコントローラ全体のダウンを引き起こさないようにするために、通信手段６１１０，バックアップ依頼手段６１２０，スケジューラ６１４０を提供するタスクなどのコントローラの管理運用に必要不可欠なタスクを最も優先して選択する。ネットワーク１０００の通信量のボトルネックが顕在化して通信不良を引き起こすことがないようにするために、他のコントローラでバックアップした場合に多量の通信を必要とするタスクが次に優先して選択される。その後、その他のタスクを選択する。各優先度においてバックアップを担当するプロセッサの負荷が限界値を越えない範囲で最大負荷となる組合せのタスクを選択する。この負荷の限界値は記憶手段６１６０に格納されており、限界負荷率と呼ぶ。この選択において記憶手段６１６０に格納されている各タスクの負荷率とこの選択における優先度とタスクの原籍に関する情報を用いる。タスクの負荷率とは着目したタスクを当該プロセッサで実行した場合の単位時間当りに必要とされる処理時間であり、あらかじめ同等の処理能力を持つプロセッサあるいは当該部分で実行したときに計測した値である。なんらかの事情で計測できないときにはあらかじめ推定し定めた値を用いる。タスクの原籍は各タスクが元々所属するコントローラあるいはその一部分を示す。なお、これらの情報は図４９に示したタスク負荷表と呼ぶ表に格納されている。また、この選択において、バックアップを担当するプロセッサがバックアップ前に実行していたタスクであろうとも優先度が低い場合にはバックアップ後に実行させないことがある。このように追い出されたタスクは、バックアップされるべきタスクに追加され、他のプロセッサでバックアップされることを待つ。

【０１１４】つづいて本手段は手順６５５０にて、選択したタスクをスケジューラ６１４０に登録し実行手段６１５０にて実行を開始させる。バックアップを担当するプロセッサから追い出されたタスクはスケジューラ６１４０から削除しその実行を停止させる。

【０１１５】つづいて本手段は手順６５６０から手順６５９０にて、バックアップされるべきタスクの状況と、

コントローラ内でバックアップを担当させていないプロセッサの有無とに応じて３通りの処理を実行する。まず、バックアップされるべきタスクに優先度の高いタスクがあり、かつ、コントローラ内の稼働中のプロセッサのすべてにバックアップを担当させる前述の手順を実施していない場合には、次に負荷率が低いプロセッサをバックアップを担当するプロセッサとして選択し、再び手順６５２０より内部バックアップ手順を繰り返す。多くのタスクあるいは多くの負荷をコントローラ内でバックアップできるように、コントローラ内の各プロセッサに対して順番にバックアップを担当させるというこの繰り返しを設けた。一方、バックアップされるべきタスクが優先度の低いタスクのみの場合と、バックアップされるべきタスクに優先度の高いタスクがあるが、コントローラ内の稼働中のプロセッサのすべてにバックアップを担当させる前述の手順を実施した場合には、コントローラの新しい実行状況をネットワーク１０００を通して放送し、バックアップされるべきタスクをコントローラ間でバックアップさせるためにバックアップ依頼手段６１２０にバックアップされるべきタスクが残っていることを前記接続を介して通知し図４５のコントローラ間のバックアップに関する手順を開始させ、本手段の処理を終了する。さらに、一方で、バックアップされるべきタスクがない場合には、コントローラの新しい実行状況をネットワーク１０００を通して放送し、本手段の処理を終了する。

【０１１６】バックアップ依頼手段６１２０は通信手段６１１０と記憶手段６１６０と内部バックアップ手段６１７０とに接続されている。この手段は、内部バックアップ手段６１７０からバックアップされるべきタスクが残っていることを通知された場合に、図４５の手順に従い、ネットワーク１０００に接続された他のコントローラあるいはそのプロセッサからバックアップの依頼先を選定しバックアップを依頼する。すべてのタスクのバックアップが終了するまでこの選定と依頼を繰り返す。この手順をバックアップ依頼手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【０１１７】この手段は手順６６１０にて、記憶手段６１６０に格納されているすべてのプロセッサの負荷状態に関する情報とを用いて、バックアップを依頼するコントローラあるいはそのプロセッサを選定する。選定基準は他のコントローラのプロセッサのなかで最も負荷の小さいことである。なお、この情報は図４９に示した外部受付順位表に格納されている。

【０１１８】つづいて本手段は手順６６２０から手順６６４０にて、さきに選定した依頼先に宛ててバックアップを依頼することを示すメッセージ６６００を発信する。このメッセージ６６００はバックアップされるべきタスクの情報を含む。メッセージ６６００はネットワーク１０００を介して送られる。そして、依頼先からの応答を待つ。応答は依頼先によってシステム全体に放送されるところの、バックアップの依頼に対する応答であることを示すメッセージ６７００である。メッセージ６７００はさらに依頼先がバックアップしたタスクとこのバックアップにより依頼先で実行されなくなったタスク（追い出されたタスク）を示す情報を含む。本手段はこのメッセージ６７００を受信するまで待ち状態に入っており、これを受信すると待ち状態が解除されて次の手順６６４２に進む。

【０１１９】つづいて本手段は手順６６４２から６６４８にて、依頼先がバックアップしたタスクの有無を判定し、依頼先がバックアップしたタスクがある場合にはバックアップの依頼の終了を判定する手順（手順６６５０から手順６６７０）に進む。依頼先がバックアップしたタスクがない場合には、さらに手順６６４４にて他のコントローラのプロセッサのすべてにバックアップを依頼したかを判定する。他のコントローラのプロセッサのすべてにバックアップを依頼していない場合には、手順６６４６にて他のコントローラのプロセッサの中で今回の依頼先に次いで負荷が小さいものを次回のバックアップ依頼先に指定し、手順６６２０に戻りバックアップの依頼を繰り返す。また、他のコントローラのプロセッサのすべてにバックアップを依頼していた場合には、バックアップが失敗している。そこでこの場合には手順６６４８に示すように縮退運転手段に縮退運転を開始しさせ、本手段はバックアップ依頼手順を終了する。

【０１２０】手順６６４２の判定にて依頼先がバックアップしたタスクがあった場合には、つづいて、本手段は手順６６６０から手順６６７０にて、依頼先からのメッセージ６７００に応じて記憶手段６１６０に格納されている各表を変更する。さらに、依頼先がバックアップしたタスクをバックアップされるべきタスクから削除し、依頼先から追い出されたタスクをバックアップされるべきタスクに追加する。その後、バックアップされるべきタスクの有無を判定し、バックアップされるべきタスクが有るときには手順６６１０に戻り依頼先の選定とバックアップの依頼を繰り返す。この依頼の繰り返しと依頼先の拒否あるいは受付により、バックアップされるべきタスクを自律的に分散してバックアップすることを可能にした。バックアップ受付手段６１３０は通信手段６１１０と記憶手段６１６０とに接続されている。この手段は、ネットワーク１０００を介して他のコントローラから送られてきたバックアップを依頼するメッセージ６６００を受け取った場合に、図４６の手順に従い、メッセージ６６００によってバックアップを依頼されたタスクをバックアップを依頼されたプロセッサの負荷状態に応じてバックアップする。この手順ではバックアップを担当するプロセッサの負荷率が限界負荷率を越えない範囲できる限り大きくなるように、バックアップを依頼され

たタスクとそのプロセッサが実行しているタスクとの入れ替えも実施する。この入れ替えはかなり大きい負荷率のタスクをバックアップするために必要な機能である。大きな負荷率のタスクをバックアップする余裕が無くとも、小さな負荷率のタスクを追い出し余裕を作れば大きな負荷率のタスクをバックアップできることが多い。追い出されたタスクはバックアップされるべきタスクとして、さらに他のプロセッサにバックアップ依頼されそこで実行されることになろう。あるいはさらにそこで入れ替えを引き起こすかも知れないが、入れ替えの度により小さな負荷率のタスクが追い出されるためバックアップできる可能性が高まってゆき、最終的にはバックアップできるであろう。この入れ替えにおいて自らのコントローラの管理運用に必要不可欠なタスクあるいは他のコントローラで実行した場合に多量の通信を必要とするタスクを追い出してしまうと、自らのコントローラのダウンやネットワーク１０００の通信量のボトルネックによる通信の不良を引き起こしたりする。これを防ぐためにバックアップ後に実行するタスクとしてそれらのタスクを優先して選択するこの手順とした。また、本手段はバックアップの結果をシステム全体に知らしめる。この手順をバックアップ受付手順と呼び、後にその詳細を具体的な例により説明するが、ここでは本手段の機能を説明するうえでその概略を説明する。

【０１２１】この手段は手順６７１０から手順６７３０にて、バックアップされるべきタスクとバックアップを担当するプロセッサで実行されている複数のタスクの中から、バックアップを担当するプロセッサがバックアップ後に実行すべきタスクを優先度に応じて選択する。このとき、バックアップにより自らのコントローラがダウンしないようにするために、通信手段６１１０，バックアップ依頼手段6120，スケジューラ６１４０を提供するタスクなどの自らのコントローラの管理運用に必要不可欠なタスクを最も優先して選択する。ネットワーク１０００の通信量のボトルネックが顕在化して通信不良を引き起こすことがないようにするために、他のコントローラで実行した場合に多量の通信を必要とするタスクが次に優先して選択される。その後、その他のタスクを選択する。各優先度においてバックアップを担当するプロセッサの負荷が限界値を越えない範囲で最大負荷となる組合せのタスクを選択する。この負荷の限界値は記憶手段６１６０に格納されており、限界負荷率と呼ぶ。この選択は図４４の内部バックアップ手順の手順６５３０と同一である。また、この選択において、バックアップを担当するプロセッサがバックアップ前に実行していたタスクであろうとも優先度が低い場合にはバックアップ後に実行させないことがある。つづいて、本手段は選択したタスクをスケジューラ６１４０に登録し実行手段６１５０にて実行を開始させる。バックアップを担当するプロセッサから追い出されたタスクはスケジューラ６１４０から削除しその実行を停止させる。

【０１２２】つづいてこの手段は手順６７４０にて、バックアップするタスクと追い出すタスクを示す情報を含むメッセージ６７００をネットワーク上に放送（ブロードキャスト）する。このメッセージ６７００により自らのコントローラの実行状態の変化を他のコントローラのすべてにに知らしめる。さらに、本手段はすべてのコントローラは各々の記憶手段６１６０，６２６０，６３６０，６４６０に格納されている実行分担表とタスク負荷表と外部受付順位表をこのメッセージ６７００によって書き換える。また前述したように、このメッセージ６７００に基づいて依頼元のバックアップ依頼手段（６２２０，６３２０，６４２０のいずれか）はバックアップされるべきタスクからバックアップされたタスクを削除し，追い出されたタスクをバックアップされるべきタスクに追加する。

【０１２３】このように１台のダウンあるいは負荷オーバしたコントローラが実行していた複数のタスクがバックアップを依頼された稼働中のコントローラの負荷に応じて分割されることにより、最終的には複数のコントローラでバックアップされる。つまり、本発明の分散制御システムは１台のコントローラのダウンあるいは負荷オーバを複数台の稼働中のコントローラが自らの負荷状況応じて分担してバックアップするものである。さらに、このバックアップにおいてネットワークの通信量増加を抑さえるために、タスクの通信量に関する属性に応じて分担してバックアップするものである。

【０１２４】この分散制御システムの物理的な構成を図４７に示す。コントローラ１，２，３，４は同一の構成であり、各々がネットワーク１０００と接続されたネットワーク接続用コネクタ４１ａ，４１ｂ，４１ｃ，４１ｄと、これらのネットワーク接続用コネクタの各々とシリアルデータ用のバス７１ａ，７１ｂ，７１ｃ，７１ｄを介して接続された４個づつのプロセッサ（ＰＲ１，ＰＲ２，ＰＲ３，ＰＲ４）７０１～７０４，７１１～７１４，７２１～７２４，７３１～７３４と、これらのプロセッサ（ＰＲ１，ＰＲ２，ＰＲ３，ＰＲ４）とバス７０ａ，７０ｂ，７０ｃ、７０ｄを介して接続されたグローバルメモリ８２ａ，８２ｂ，８２ｃ，８２ｄと、これらのプロセッサ（ＰＲ１，ＰＲ２，ＰＲ３，ＰＲ４）と接続された入出力処理装置（Ｉ／Ｏ）８３ａ，８３ｂ，８３ｃ，８３ｄと、これらのプロセッサに接続され故障の通知を行うバス７０ｂを含む。各コントローラにはＤＣ１，ＤＣ２，ＤＣ３，ＤＣ４という名前を付けた。各コントローラ内の各プロセッサにはＰＲ１，ＰＲ２，ＰＲ３，ＰＲ４という名前を付けた。ここで、各プロセッサはその中に記したタスクを実行しているものとした。ここでタスクＮＣ１，ＮＣ２，ＮＣ３，ＮＣ４はネットワーク１０００を介したコントローラ間のメッセージの送受信を制御するタスクであり、図４３の通信手段６１１

０，６２１０，６３１０，６４１０を提供する。また、タスクＣＭ１，ＣＭ２，ＣＭ３，ＣＭ４は各コントローラ１，２，３，４でのタスクの実行を制御するタスクであり、図４３のバックアップ依頼手段６１２０，６２２０，６３２０，６４２０と、バックアップ受付手段６１３０，６２３０，６３３０，６４３０を提供する。頭文字がＴで始まるタスクは各コントローラが実行する制御演算などであり、図４３の制御手段を提供する。ＯＳ１１などのＯＳで始まるタスクはバックアップ関係の機能を組み込んだリアルタイムオペレーティングシステムであり、各プロセッサで実行されている。これら各々が図４３の内部バックアップ手段６１７０，6270，６３７０，６４７０を提供する。またこれらのリアルタイムオペレーティングシステムの各々が独立に各々のプロセッサに関するスケジューリングを実施している。すなわち、各々が各コントローラのスケジューラ６１４０，６２４０，６３４０，６４４０を各プロセッサに関する部分に分割して提供している。

【０１２５】また、各プロセッサ内の故障制御回路６８２０（図8，図１０，図１２参照）は他のプロセッサの故障制御回路と協同してバス７０ｂを通信路として用いることにより切り離し手段を提供する。例えば図４７のプロセッサ７０２が故障した場合には、プロセッサ７０２の故障制御回路が故障を検出し、他のプロセッサ７０１，７０３，７０４に対してバス７０ｂを通してプロセッサ７０２の故障を通知し、同時にプロセッサ７０２の汎用エンジン(ＧＥ)の動作を停止さる。さらに、プロセッサ７０１，７０３，７０４の各故障制御回路は各々のプロセッサＰＲで実行されている内部バックアップ手段を提供するタスクＯＳ１１，ＯＳ１３、ＯＳ１４に内部バックアップ手順を開始させる。このように図４３の切り離し手段６１８０における機能は物理的には４つのプロセッサに分担されている。

【０１２６】さらに、図４７の各コントローラのなかのすべてのプロセッサがランアダプタ（ＬＡ）を持つため、いずれのプロセッサも通信手段６１１０，６２１０，6310，６４１０として動作可能である。従来技術では通信関係の手段が冗長化されておらず信頼性が不充分であったが本実施例の構成により冗長化され信頼性が向上する。各コントローラ内の４つのランアダプタ（ＬＡ）のなかで、実際に通信手段６１１０，６２１０，６３１０，６４１０となるものはタスクＮＣ１，ＮＣ２，ＮＣ３，ＮＣ４が実行されているプロセッサのランアダプタ（ＬＡ）である。つまり通信手段として動作中のランアダプタ(ＬＡ)が故障して通信が不通になっても、タスクＮＣ１，ＮＣ２，ＮＣ３，ＮＣ４をコントローラ内でバックアップすることで通信を回復することができる。また、従来技術ではプロセッサあるいはプロセッサがローカルメモリを持った場合、そのプロセッサあるいはプロセッサがダウンすると、ローカルメモリに蓄えられた制御用の学習結果のデータなどにアクセスできないという問題があった。そこでローカルメモリ(ＬＭ)６８１６をデュアルポートメモリで構成し、バス６８１９のほかにバス７０ａにも接続した。この接続により他のプロセッサからバス７１ａを介してローカルメモリ(LM)６８１６へアクセスできるようにした。コントローラ内の４つのプロセッサ（ＰＲ１，ＰＲ２，ＰＲ３，ＰＲ４）のローカルメモリ(ＬＭ)とグローバルメモリ（ＧＭ）によって図３４の記憶手段６１６０，６２６０，６３６０，６４６０を構成している。

【０１２７】以下では本実施例のバックアップ方法について詳細に例示する。図４８は図４７の動作状態における実行分担表であり、各記憶手段６１６０，６２６０，６３６０，６４６０に格納されている。この表の内容はプロセッサを特定可能な名前と、そのプロセッサが実行しているタスクの名称と、それらのタスクによってプロセッサが被る負荷率と、各コントローラ内での各プロセッサに負荷の軽い順に１番から番号を付けた順位（内部負荷順位）である。本例ではコントローラ名とプロセッサ名を連結しプロセッサを特定可能な名称としている。このほかの特定する方式として各々異なる番号を与えるものがある。このプロセッサを特定するために用いる方式は本発明の主旨とは関係なくどのような方式でもよい。本表を用いて内部バックアップ手段６１７０，６２７０，６３７０，６４７０は、バックアップされるべきタスクをダウンあるいは負荷オーバしたプロセッサ名から検索し、コントローラ内で負荷の小さいプロセッサから大きいプロセッサへ順番にバックアップを担当させるために内部負荷順位を検索する。また、バックアップ受付手段６１３０，６２３０，６３３０，６４４０がバックアップを依頼されたプロセッサ名からそのプロセッサが実行しているタスク名を検索するために実行分担表を用いる。なお、内部負荷順位が無くても、内部バックアップ手段はプロセッサ名と負荷率をもとに内部負荷順位と同等の情報を生成し上記の処理を実行可能であるが、バックアップの途中でこの情報を生成するためバックアップを終了するまでの時間が長くなる。バックアップを終了するまでの時間を短縮するために、本実施例では通常の稼働状態の時に内部負荷順位を生成しこの実行分担表に記憶しておく方式にした。

【０１２８】図４９は図４７の動作状態におけるタスク負荷表であり、各記憶手段６１６０，６２６０，６３６０，６４６０に格納されている。この表の内容はタスクの名前と、そのタスクを実行したときの負荷率と、バックアップに関係する属性と、タスクがどのプロセッサのものかを示す原籍である。属性は４つあり、原籍のプロセッサにおいて必要不可欠であるが他のプロセッサでバックアップする必要のない（あるいはしてはならない）タスクであることを示す“fixed”と、コントローラの管理運用に必要なタスクであることを示す“internall

y”と、通信量が多いことを示す“communicative”と、原籍以外のコントローラで実行してもよいタスクであることを示す“somewhere”である。よって、他のプロセッサでバックアップされるべきタスクは属性が“fixed”でないものである。これらをあるプロセッサにおける優先度として扱い優先度が高いものから降順に並べると、最優先のものはこのプロセッサが原籍である“fixed”、次に優先されるものはこのプロセッサが原籍である“internally”、さらに次に優先されるものはこのプロセッサが原籍である“communicative”、優先度の最も低いものはこのプロセッサが原籍である“somewhere”とこのプロセッサが原籍でない“fixed”，“internally”，“communicative”と“somewhere”である。ただし、原籍以外のプロセッサに属性が“fixed”や“internally”のタスクをバックアップさせることはコントローラあるいはシステムの不具合を生じるため、バックアップ手順の中でこれを防いでいる。すなわち、原籍のプロセッサあるいはそれを含むコントローラ（原籍のコントローラと呼ぶ）にて最優先でバックアップされるようにしている。内部バックアップ手段６１７０，６２７０，６３７０，６４７０と、バックアップ受付手段６１３０，６２３０，６３３０，６４４０がバックアップをされるべきタスクとバックアップを担当するプロセッサが実行していたタスクとから各々のタスクの優先度と原籍と負荷率を基準としてバックアップ後に実行すべきタスクを選択する時に、タスク名からそれらの内容を知るためにこのタスク負荷表を用いる。

【０１２９】図５０は図４７の動作状態における外部受付順位表でであり、各記憶手段6160，６２６０，６３６０，６４６０に格納されている。本表はバックアップ依頼手段６１２０，６２２０，６３２０，６４２０が他のコントローラのプロセッサの中から、最小負荷率のプロセッサの名称を最初に検索し、つづいて必要ならば順番に負荷の大きいものの名称を検索するために用いる。この表はプロセッサ名を負荷率をキーとして昇順に並べたものである。よって、実行分担表に本表の情報も含まれているが、実行分担表で最も負荷率の小さいプロセッサを検索するためには相当な時間がかかる。そこで、検索時間短縮を目的として本表を設けた。本表では先頭のプロセッサ名を読みだすだけですべてのプロセッサの中で最小負荷率のプロセッサ名を知ることができる。それがバックアップを依頼するコントローラに属するときは次の順位を読みだす。この繰り返しで他のコントローラのプロセッサで最も負荷が小さいものを選択できる。本表はダウンあるいは負荷オーバやバックアップの度に書き換えるが、一度並べ帰られているので削除，２分検索，挿入の３ステップでこの表の昇順を維持する。このため、この外部受付順位表を用いた場合にバックアップ時に実行分担表の内容をソーティングを実施する必要がなく処理が高速になる。すなわち、バックアップに要する時間が短縮できる。

【０１３０】ここで、実行分担表とタスク負荷表と外部受付順位表の内容の関係を説明する。プロセッサＤＣ１．ＰＲ１が実行しているタスクは実行分担表よりタスクOS11，ＮＣ１，ＣＭ１，Ｔ１１である。このときの負荷率はタスク負荷表からタスクＯＳ１１が５%，ＮＣ１が２６%，ＣＭ１が２８%，Ｔ１１が１３%であるからこれらの和である７２%となる。実行分担表の例示で省略されたプロセッサの負荷率はこの値より充分大きいとして、プロセッサＤＣ１．ＰＲ１の負荷率は昇順で３番目であるから、外部受付順位表では順位が３番目になっている。

【０１３１】ここで、本実施例のバックアップ手順の理解の助けとして、図５１に示すようにコントローラ１のプロセッサ（ＤＣ１．ＰＲ１）７０１がダウンした場合において、タスクＮＣ１，ＣＭ１，Ｔ１１が分割してバックアップされる過程とそれにより追い出されたタスクがさらにまた他のプロセッサによってバックアップされる過程を説明する。ここで前述の限界負荷率は９０%に定められているものとする。

【０１３２】（ＦＤ0）コントローラ１のプロセッサ７０１のプロセッサ６８１１が故障したものとする。この故障により通信手段６１１０とバックアップ依頼手段６１２０とバックアップ受付手段６１３０がダウンし、スケジューラ６１４０と制御手段６１５０の各一部分がダウンした。

【０１３３】（ＦＤ１）コントローラ１の切り離し手段６１８０がプロセッサ７０１のダウンを検出すると、このプロセッサを停止させる。同時に内部バックアップ手段６１７０にプロセッサ７０１のダウンを通知し、内部バックアップ手段６１７０に図４６の内部バックアップ手順を開始させる。

【０１３４】より詳細には、プロセッサ７０１の故障制御回路（ＦＤ）６８２０が汎用エンジン（ＧＥ）６８１１の故障を検出すると、バス７０ｂを通して他のプロセッサの故障制御回路（ＦＤ）にプロセッサ７０１の故障を通知する。その後プロセッサ７０１を停止させる。プロセッサ７０２，７０３，７０４の故障制御回路（ＦＤ）６８２０は故障の通知を受けると各々のプロセッサ６８１１に対してバス７０ａの割り込み要求信号線を用いて割り込み処理を要求する。この割り込み処理が、内部バックアップ手段６１７０を提供するタスクＯＳ１２，ＯＳ１３，ＯＳ１４に内部バックアップ手順を開始させる。なお、停止されたプロセッサのローカルメモリ（ＬＭ）６８１６はバス７０ａと接続されているため、このプロセッサが停止していても他のプロセッサからアクセス可能である。この接続はタスクをバックアップし再開するときに必要なデータをプロセッサが停止していても取得できるようにしたものである。

【０１３５】以下、内部バックアップ手段６１７０が実

行する図４６に示した内部バックアップ手順について説明する。

【０１３６】(ＯＳ０)内部バックアップ手段６１７０が起動される。詳細には稼働中のプロセッサのすべてにおいて内部バックアップを司るタスクが各々独立に図４６の内部バックアップ手順を実行する。図５１の例示ではプロセッサ(ＤＣ１．ＰＲ２)７０２でタスクＯＳ１２が内部バックアップ手順を実行する。プロセッサ(DC1．ＰＲ３）７０３でタスクＯＳ１３が内部バックアップ手順を実行する。プロセッサ（ＤＣ１．ＰＲ４）７０４でタスクＯＳ１４が内部バックアップ手順を実行する。バックアップを担当しないタスクが自ら停止し、自動的にバックアップを実施するものが決定する。プロセッサの間にバックアップに関する特定の依存関係がないためコントローラ内を均一のハードウェア及び均一のソフトウェア体形とすることが可能となり、コントローラの構築が容易になる。このように図４６の内部バックアップ手順を構築した。次のステップでこの決定方法を説明する。

【０１３７】（ＯＳ１）手順６５１０：内部バックアップ手段６１７０は上記の稼働中のプロセッサのなかで負荷が最も小さいプロセッサをバックアップを実施するプロセッサに決定する。

【０１３８】具体的には前記のタスクの各々が実行分担表からダウンしたプロセッサの内部負荷順位と自らの内部負荷順位を読みだす。ダウンしたプロセッサの負荷が最も小さいのならば内部負荷順位は前述のように１番である。よって、その内部負荷順位が１ならば稼働中のプロセッサの中で負荷が最も小さいものは内部負荷順位が２番目である。この場合には、自らの内部負荷順位が２番であることを判定することにより、自らがバックアップを担当するか否かを定める。内部負荷順位が２番ならばバックアップを担当することになり、手順６５２０に進む。もし、内部負荷順位が２番でなければ内部バックアップ手順を終了し、バックアップを担当しない。一方、ダウンしたプロセッサの負荷より稼働中のプロセッサの負荷が小さい場合には、稼働中のプロセッサの中で負荷が最も小さいものは内部負荷順位が１番目である。この場合には、自らの内部負荷順位が１番であることを判定することにより、自らがバックアップを担当するか否かを定める。内部負荷順位が１番ならばバックアップを担当することになり、手順６５２０に進む。もし、内部負荷順位が１番でなければ内部バックアップ手順を終了し、バックアップを担当しない。

【０１３９】図５１の例示ではダウンしたプロセッサ(ＤＣ１．ＰＲ１)７０１の内部負荷順位が図４８の実行分担表に示すとおり２番であり、プロセッサ(ＤＣ１．ＰＲ２)７０２の内部負荷順位が４番であるため、タスクＯＳ１２は手順６５１０で内部バックアップ手順を終了する。また、プロセッサ（ＤＣ１．ＰＲ３）７０３の内部負荷順位が３番であるため、タスクＯＳ１３は手順６５１０で内部バックアップ手順を終了する。一方、プロセッサ（ＤＣ１．ＰＲ４）７０４の内部負荷順位は１番であるため、タスクＯＳ１４は手順６５１０から手順６５２０に進み、内部バックアップ手順を継続する。よって、プロセッサ（ＤＣ１．ＰＲ４）７０４がまず初めにバックアップを実施する。

【０１４０】（ＯＳ２）手順６５１５：内部バックアップ手段６１７０はダウンにより実行できなくなったタスクのうちバックアップする必要の有るものをタスクキューに登録する。詳細にはまず受付用タスクキューを初期化する。そして、実行分担表からダウンしたプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そして、属性がバックアップする必要がないこと、あるいは、バックアップしてはならないことを示す“fixed”のものを除外して、バックアップされるべきタスクとして受付用タスクキューに登録する。図５１の例示ではダウンしたプロセッサ(ＤＣ１．ＰＲ１)７０１の名前ＤＣ１．ＰＲ１に対応するタスク名であるＯＳ１１とＮＣ１とCM1とＴ１１が実行分担表から読みだされ、タスク負荷表から各々の負荷率５％，２６％，２８％，１３％と属性“fixed”，“communicative”，“somewhere”が読みだされる。そして、タスクの属性が“fixed”であるタスクＯＳ１１に関する情報を除いて、これらの内容が受付用タスクキューに登録される。

【０１４１】（ＯＳ３）手順６５２０：内部バックアップ手段６１７０はバックアップを担当するプロセッサが実行しているタスクを受付用タスクキューに追加する。詳細には実行分担表からバックアップを担当するプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そしてそれらのタスクを受付用タスクキューに追加する。同時にそれらの負荷率及び属性も追加する。

【０１４２】この手順以降はコントローラ内部のすべてのプロセッサがバックアップを担当するまで、あるいは、バックアップされるべきタスクがなくなるまで繰り返し実行される。図５１の例示における今回の実行では、バックアップを担当するプロセッサ７０４の名前ＤＣ１．ＰＲ４に対応するタスク名であるＯＳ１４とＴ１７とＴ１８が実行分担表から読みだされ、タスク負荷表から各々の負荷率５％，２５％，２６％と属性“fixed”，“internally”，“communicative”が読みだされる。これらが受付用タスクキューに追加される。

【０１４３】（ＯＳ４）手順６５３０：内部バックアップ手段６１７０はバックアップを担当するプロセッサがバックアップ後に実行すべきタスクを受付用タスクキューから選択する。具体的には、受付用タスクキューをタスクの属性による優先順位と負荷率で並べ変えて、優先順位の高いタスクのなかで限界負荷率に最も近くなる組合せを選択し、さらにすべてを選択しても負荷率に余裕

があるならば次の優先度のタスクのなかでその余裕に最も近くなる組合せを選択する。何れかの優先順位のタスクの中で選択できないものが生じるまで、この組合せと選択を繰り返す。図５１の例示では受付用タスクキューが図５２に示した内容になり、タスクＯＳ１４とＮＣ１とＣＭ１とＴ１７が最終的に選択さる。ここで、バックアップされるタスクはＮＣ１とＣＭ１であり、バックアップされないタスクはＴ１１であり、追い出されるタスクはＴ１８である。

【０１４４】（ＯＳ５）手順６５４０：前手順により選択されたタスクを受付用タスクキューから削除する。具体的には、受付用タスクキューから前手順で選択されたタスクの名称とその負荷率と属性を削除する。図５１の例示ではタスクＯＳ１４とＮＣ１とＣＭ１とＴ１７が最終的に選択さた。当手順では、これらのタスクを受付用タスクキューから削除する。受付用タスクキューに残されたタスクは、バックアップされなかったタスクＴ１１と追い出されたタスクＴ１８である。

【０１４５】（ＯＳ６）手順６５５０：手順６５３０にて選択されたタスクをスケジューラ６１４０に登録し実行を開始させる。具体的には、バックアップを担当するプロセッサのスケジューリングを担当するタスクに対してバックアップされるタスクを追加登録し、追い出されるタスクの登録を削除する。図５１の例示ではバックアップを担当するプロセッサ７０４のスケジューリングを担当するタスク、すなわち、リアルタイムオペレーティングシステムであるタスクＯＳ１４に、バックアップされるタスクであるタスクＮＣ１とＣＭ１を追加登録し実行を開始させる。また、追い出されるタスクであるタスクＴ１８をスケジューラから削除し実行を停止させる。ここにおいて、プロセッサ７０１の故障によりダウンしていた通信手段６１１０とバックアップ依頼手段６１２０とバックアップ受付手段6130が回復された。以降、コントローラ間のバックアップや制御タスクの実行に必要なネットワーク１０００との通信が実行できる。

【０１４６】（ＯＳ７）手順６５６０：当手順はバックアップの状況に応じて内部バックアップ手順を３通りに分岐させる。一つは完全にバックアップが終了したのでバックアップに関する手順を終了させる処理への分岐である。二つは内部でのバックアップを完了しコントローラ間におけるバックアップを開始させる処理への分岐である。三つは内部でのバックアップを繰り返す処理への分岐である。手順6560の中でこれらの分岐を手順６５６２，６５６４，６５６６が実施している。手順６５６２はバックアップすべきタスク、すなわち、受付用タスクキューに残っているタスクの中で優先度の高いタスクが有るか否かを判定し、有るならば処理は手順６５６４へ進む。無いならば処理は手順６５６６へ進む。手順６５６４はコントローラ内のすべてのプロセッサにバックアップを担当させたか否かを判定し、すべてに担当させた後ならば処理は手順６５７４以降のコントローラ間におけるバックアップを開始させる処理へ分岐する。すべてに担当させていないならば処理は手順６５８０以降の内部でのバックアップを繰り返す処理へ分岐する。手順６５６６はバックアップすべきタスクが残っているか否かを、すなわち、受付用タスクキューにタスクが残っているか否かを判定し、タスクが残っているならば処理は手順６５７４以降のコントローラ間におけるバックアップを開始させる処理へ分岐する。タスクが残っていないならば処理は手順６５７２以降のバックアップに関する手順を終了させる処理へ分岐する。

【０１４７】図５１の例示において、今回は、属性が“communicative”であり優先度が高いタスクＴ１１が受付タスクキューに残っており、尚且つ、まだプロセッサ704にバックアップを担当させたのみであるため、手順６５８０以降の内部でのバックアップを繰り返す処理へ分岐する。

【０１４８】（ＯＳ８）手順６５８０：今回バックアップを担当したプロセッサの次に負荷が小さいものを次回のバックアップを担当するプロセッサに選定し、次回のバックアップを実施するために手順６５２０へ戻る。詳細には今回バックアップを担当したプロセッサの内部受付順位とダウンしたプロセッサの内部受付順位を実行分担表から読みだす。前者の値に１を加え次の内部受付順位を求める。この値が後者の受付順位であった場合にはさらに１を加えてダウンしたプロセッサを指定してしまうことを防ぐ。コントローラ内で内部受付順位がこの値であるプロセッサを実行分担表から求める。求めたプロセッサを次回にバックアップを担当するものとして選択する。図５１の例示では今回バックアップを担当したプロセッサ７０４の内部受付順位が１番であり、ダウンしたプロセッサの内部受付順位が２番であるため、次回にバックアップを担当するプロセッサの内部受付順位は３番になる。よって実行分担表においてコントローラ１内で内部受付順位が３番であるプロセッサ７０３が次回のバックアップを担当するものとして選択される。ここから２回目の内部バックアップにはいる。すでに説明した各手順は図５１の例示に関する説明のみを記述する。

【０１４９】（ＯＳ９）手順６５２０：図５１の例示における今回の実行では、バックアップをプロセッサ７０３が担当する。そこで、その名前ＤＣ１．ＰＲ３に対応するタスク名であるＯＳ１３とＴ１５とＴ１６が実行分担表から読みだされ、タスク負荷表から各々の負荷率５％，２８％，４１％と属性“fixed”，“somewhere”，“communicative”が読みだされる。これらが受付用タスクキューに追加される。

【０１５０】（ＯＳ１０）手順６５３０：図５１の例示では受付用タスクキューが図５３に示した内容になり、タスクＯＳ１３とＴ１５とＴ１６とＴ１１が最終的に選択さる。ここで、バックアップされるタスクはＴ１１で

あり、バックアップされないタスクはＴＩ８であり、追い出されるタスクは無い。

【０１５１】（ＯＳ１１）手順６５４０：図５１の例示ではタスクＯＳ１３とＴ１５とT16とＴ１１が最終的に選択された。当手順では、これらのタスクを受付用タスクキューから削除する。受付用タスクキューに残されたタスクは、バックアップされなかったタスクＴ１８のみである。

【０１５２】（ＯＳ１２）手順６５５０：図５１の例示ではバックアップを担当するプロセッサ７０３のスケジューリングを担当するタスク、すなわち、リアルタイムオペレーティングシステムであるタスクＯＳ１３に、バックアップされるタスクであるタスクＴ１１を追加登録し実行を開始させる。また、追い出されるタスクが無いため、スケジューラから削除され実行を停止させられるものは無い。

【０１５３】（ＯＳ１３）手順６５６０：図５１の例示において、今回は、属性が“somewhere”であり優先する必要がないタスクＴ１８のみが受付タスクキューに残っている。すなわち、この状態は優先度の高いタスクは残っていないが、バックアップすべきタスクが残っている状態である。よって、当手順は処理を手順６５７４以降のコントローラ間におけるバックアップを開始させる処理へ分岐させる。

【０１５４】（ＯＳ１４）手順６５７４：新しい実行状態を示すメッセージ６７００をネットワーク１０００に放送（ブロードキャスト）する。このメッセージにより他のコントローラはプロセッサのダウンと、そのバックアップを担当したプロセッサの新しい実行状態を知ることができる。このメッセージ６７００の内容に合わせて、各コントローラは各々の記憶手段に格納している実行分担表とタスク負荷表と外部受付順位表を修正する。図５１の例示において、メッセージ６７００はダウン情報としてプロセッサ７０１のダウンを示す。さらに、新しい実行状況としてプロセッサ７０３が実行しているタスクＯＳ１３，Ｔ１５，Ｔ１６，Ｔ１１と、プロセッサ７０４が実行しているタスクＯＳ１４，ＮＣ１，ＣＭ１，Ｔ１７とを示す。修正された実行分担表と外部受付順位表を図５５及び図５６に示す。

【０１５５】（ＯＳ１５）手順６５９０：コントローラ間でのバックアップを実施するために、バックアップ依頼手段６１２０を起動する。詳細には残ったバックアップされるべきタスクを受付用タスクキューからバックアップ依頼手段６１２０が用いる依頼用タスクキューに移し、受付用タスクキューを空にする。そして、バックアップ依頼手段６１２０にバックアップされるべきタスクが残っていることを通知する。この通知はリアルタイムオペレーティングシステムのタスク間通信機能を利用して行われてもよい。あるいは、ナットワーク１０００を介した自コントローラから自コントローラ宛のメッセージ通信を利用して行われてもよい。

【０１５６】（ＯＳ１６）以上にて内部バックアップ手順を終了する。

【０１５７】続いて、コントローラ間でバックアップ手順に移行し、タスクＴ１８を他のコントローラでバックアップする。

【０１５８】（ＣＭ１）コントローラ１のバックアップ依頼手段６１２０が起動され、図４５のバックアップ依頼手順の実行を開始する。

【０１５９】（ＣＭ２）手順６６１０：バックアップ依頼手段６１２０が外部受付順位表を用い、ダウンしたプロセッサを吹く不コントローラとは異なる他のコントローラのプロセッサの中で負荷が最も小さいものをバックアップの依頼先に決定する。詳細には外部受付順位表から順位が１番のプロセッサ名を読みだす。そのプロセッサが他のコントローラに所属するならばそこで読みだしを停止し、そのプロセッサをバックアップの依頼先にする。もし、ダウンしたプロセッサと同一のコントローラに所属しているならば、再び外部受付順位表から次の順位のプロセッサ名を読みだす。そして、他のコントローラに所属しているか否かを判定する。他のコントローラに所属するプロセッサが読みだされるまで以上を繰り返す。図５１の例示ではダウンしたプロセッサを含むコントローラ内でのバックアップを終了しているため、図５０の外部受付順位表は図５６に変更されている。この図５６の外部受付順位表から順位１番のプロセッサ名を読みだす。この名前はDC2．ＰＲ２であり、このプロセッサはコントローラＤＣ２に属する。ダウンしたプロセッサを含むコントローラはＤＣ１であるため、読みだされたプロセッサはダウンしたプロセッサを含むコントローラとは異なったコントローラに属する。よって次の順位のプロセッサ名を読みだす処理に戻らずに、このプロセッサ（ＤＣ２．ＰＲ２）７１２をバックアップを依頼するプロセッサに決定する。このプロセッサを依頼先と呼ぶ。

【０１６０】（ＣＭ３）手順６６２０：バックアップ依頼手段６１２０がバックアップの依頼であることを示す情報と、依頼用タスクキューに登録されているすべてのタスク名を示す情報とを含むメッセージ６６００を作成し、依頼先に宛てて発信する。このメッセージを依頼メッセージと呼ぶ。図５１の例示では依頼メッセージ６６００はタスクＴ１８のバックアップを依頼するものであり、依頼先であるプロセッサ７１２に宛ててネットワーク１０００を介して送付される。

【０１６１】（ＣＭ４）手順６６３０：バックアップ依頼手段６１２０はバックアップを依頼したメッセージ６６００に対する回答としてバックアップの依頼先から放送されるメッセージ６７００を待つ。

【０１６２】以後、メッセージ６７００が放送されるまでバックアップ依頼手段６１２０はバックアップ依頼手

順を中断して、待ち状態となる。ここからは依頼先のバックアップ受付手段６２３０の動作について説明する。

【０１６３】（ＣＭ５）依頼先のプロセッサ７１２を含むコントローラ２に於て、メッセージ６５００を通信手段６２１０が受け取り、バックアップ受付手段６２３０を起動する。バックアップ受付手段６２３０は図４６のバックアップ受付手順を開始する。ここで、バックアップ受付手段６２３０はそれ自身を含むコントローラの記憶手段６２６０に格納されている受付用タスクキューとタスク負荷表を用いる。

【０１６４】（ＣＭ６）手順６７１０：バックアップ受付手段６１３０がメッセージ６６００によって依頼されたタスクとバックアップを依頼されたプロセッサが実行しているタスクを受付用タスクキューに登録する。詳細にはバックアップ受付手段６２３０はまず記憶手段６２６０内の受付用タスクキューを初期化する。続いて、メッセージ６６００から依頼されたタスクの名称をとりだし、記憶手段6260内のタスク負荷表からそのタスク名に対応する負荷率と属性と原籍を読みだし、先に初期化した受付用タスクキューに登録する。さらに、記憶手段６２６０内の実行分担表から依頼先のプロセッサの名前に対応するタスク名を読みだし、タスク負荷表からそれらのタスクに対する負荷率と属性を読みだす。そしてそれらのタスクを受付用タスクキューに追加する。同時にそれらの負荷率及び属性も追加する。

【０１６５】図５１の例示では、メッセージ６６００によりバックアップを依頼されたタスクの名称Ｔ１８を受付用タスクキューに登録する。同時に、タスク負荷表からその名称に対応した負荷率２６％と属性“somewhere”と原籍ＤＣ１．ＰＲ４を読みだしこれらを登録する。さらに、バックアップを依頼されたプロセッサ７１２が実行しているタスクの名称として、そのプロセッサ名ＤＣ２．ＰＲ２に対応するタスク名ＯＳ２２とＴ２１とＴ２２を実行分担表から読みだす。さらに、タスク負荷表から各々のタスクに対応した負荷率５％，１８％，３５％と属性“fixed”，“communicative”，“somewhere”と原籍ＤＣ２．ＰＲ２，ＤＣ２．ＰＲ２，ＤＣ２．ＰＲ２とを読みだす。そして、これらのタスク名と負荷率と属性と原籍を受付用タスクキューに追加する。

【０１６６】（ＣＭ７）手順６７２０：バックアップ受付手段６１３０はバックアップを依頼されたプロセッサがバックアップ後に実行すべきタスクを受付用タスクキューから選択する。具体的には、受付用タスクキューをタスクの属性による優先順位と負荷率で並べ変えて、優先順位の高いタスクのなかで限界負荷率に最も近くなる組合せを選択し、さらにすべてを選択しても負荷率に余裕があるならば次の優先度のタスクのなかでその余裕に最も近くなる組合せを選択する。何れかの優先順位のタスクの中で選択できないものが生じるまで、あるいは、全てのタスクが選択されるまでこの組合せと選択を繰り返す。図５１の例示では受付用タスクキューが図５４に示した内容になり、タスクＯＳ２２とＴ２１とＴ２３とＴ１８が最終的に選択さる。ここで、バックアップされるタスクはＴ１８であり、バックアップされないタスクは無い、追い出されるタスクも無い。すなわち、受付用タスクキュー内の全てのタスクが選択された。

【０１６７】（ＣＭ８）手順６７３０：バックアップ受付手段６１３０は前手順により選択されたタスクをスケジューラ６２４０に登録し、それらの実行を開始する。具体的には、バックアップされるタスクをスケジューラ６２４０に追加して登録し、その実行を開始させる。また追い出されるタスクの登録をスケジューラ６２４０から削除し、その実行を停止させる。図５１の例示ではタスクＴ１８を登録し、実行を開始させる。登録を削除するものはない。

【０１６８】（ＣＭ９）手順６７４０：バックアップ受付手段６１３０はバックアップを依頼されたプロセッサの新しい実行状態を示すメッセージ６７００をネットワーク１０００に放送（ブロードキャスト）する。このメッセージ６７００はバックアップの依頼に対する回答であることを示す情報と、バックアップ受付手順が終了したときにバックアップを依頼されたプロセッサが実行しているタスクの情報と、バックアップを依頼されたタスクの中でバックアップしたタスクを示す情報と、バックアップを依頼されたプロセッサから追い出されてその実行を停止されたタスクを示す情報を含む。このメッセージにより他のコントローラはバックアップを依頼されたプロセッサの新しい実行状態を知り、各々の記憶手段に格納している実行分担表とタスク負荷表と外部受付順位表を修正する。また、このメッセージ６７００によりバックアップの依頼元であるバックアップ依頼手段６１２０は待ち状態から抜け出し、バックアップ依頼手順を再開する。図５１の例示において、メッセージ６７００は新しい実行状況としてプロセッサ７１２が実行しているタスクＯＳ２２，Ｔ２１，Ｔ２２，Ｔ１８を示す。バックアップされたタスクとしてタスクＴ１８を示す。追い出されたタスクとしては、該当するものがないため、追い出されたものがないことを示す。

【０１６９】（ＣＭ１０）バックアップ受付手段６２３０がバックアップ受付手順を終了する。

【０１７０】以降、メッセージ６７００により処理を再開したバックアップ依頼手順6120の処理について説明する。

【０１７１】（ＣＭ１１）手順６６４２：バックアップ依頼手段６１２０がメッセージ6700からバックアップされたタスクを示す情報をとりだす。そして、バックアップされたタスクがあるか否かを判定する。もし、バックアップされたタスクがないならば手順６６４４にて全てのプロセッサにバックアップを依頼したか否かを判定する。さらにもし、すべてのプロセッサに依頼していない

ならば手順６６４６にて外部負荷順位表を参照して、今回の依頼先に次いで負荷が小さいプロセッサを依頼先に選定する。そして、再び手順６６２０以降によりバックアップの依頼を繰り返す。手順６６４４にて、もし全てのプロセッサに依頼した後であったならば、バックアップが失敗したため、縮退運転手段を起動し縮退運転を行う。さて、手順６６４２に戻る。ここで、バックアップされたタスクが存在するならば手順６６５０へ進む。

【０１７２】図５１の例示ではタスクＴ１８がバックアップされたため、手順６６５０へ進む。

【０１７３】（ＣＭ１２）手順６６５０：バックアップ依頼手段６１２０はメッセージ6700の内容に応じて記憶手段６１６０内の実行分担表とタスク負荷表と外部受付順位表を変更する。図５１の例示では図５５の実行分担表のマイクとコンピュータＤＣ２．ＰＲ２の実行しているタスクの欄にタスクＴ１８を追加する。同じく負荷率の欄を５８％の負荷率から８４％に変更する。コントローラ２（ＤＣ２）関係の内部負荷順位を変更し、プロセッサＤＣ２．ＰＲ１を２番に、プロセッサＤＣ２．ＰＲ２を４番に、プロセッサＤＣ２．ＰＲ３を１番に、プロセッサＤＣ２．ＰＲ４を３番にする。外部受付順位表では受付順位１番のプロセッサＤＣ２．ＰＲ２が負荷率８４％となったためそれを後ろに回し、負荷率８４％以下のプロセッサの順位を１番ずつ繰り上げる。したがって、外部受付順位が１番のプロセッサはＤＣ３．ＰＲ１になる。

【０１７４】（ＣＭ１３）手順６６６０：バックアップ依頼手段６１２０は依頼用タスクキューからメッセージ６７００に示されたところのバックアップされたタスクを削除する。また、依頼用タスクキューにメッセージ６７００に示されたところの追い出されたタスクを追加する。この操作は１回のバックアップ依頼で全てのバックアップが終了しなくてもよいように設けた。すなわち、１回のバックアップでバックアップできなかったタスク、あるいは、負荷率を上げるために追い出されたタスクがある場合には、依頼用タスクキューにタスクが残り、これを今回の依頼先と異なった依頼先に次回に依頼できるようにしたものである。図５１の例示では依頼用タスクキューからタスクＴ１８が削除され、依頼用タスクキューが空になる。

【０１７５】（ＣＭ１４）手順６６７０：バックアップ依頼手段６１２０は依頼用タスクキューにタスクが残っているか否かを判定する。もし残っているならば再び手順６６１０に戻り新規に依頼先を決定し、バックアップを依頼する。また、もし、依頼用タスクキューにタスクが残っていないならば、全てのタスクのバックアップを終了したことになる。このときバックアップ依頼手段６１２０はバックアップ依頼手順の終了に進む。

【０１７６】（ＣＭ１５）バックアップ依頼手段６１２０はバックアップ依頼手順を終了する。

【０１７７】以上のステップにより本実施例の分散制御システムは１台のプロセッサ(DC1．ＰＲ１）７０１のダウンに対して、そこで実施されていたタスクＮＣ１，ＣＭ１，Ｔ１１を自律的に分割してそれぞれをコントローラ１のプロセッサ（ＤＣ１．ＰＲ４）７０４，コントローラ１のプロセッサ（ＤＣ１．ＰＲ３）７０３，コントローラ２のプロセッサ（ＤＣ２．ＰＲ２）７１２においてバックアップする。図５１の例示ではダウンしたプロセッサを持つコントローラと、バックアップを依頼するコントローラが同じ場合を示した。しかし、本発明の主旨によればコントローラ１、２、３、４の各々がすべてのコントローラの実行状態を記述している実行分担表を持っているために、あるコントローラのダウンにおいて他の一つのコントローラがさらに他のもう一つのコントローラに対してバックアップを依頼することが可能である。具体的には図５１の例示において、コントローラ１のプロセッサ７０４でバックアップされたタスクＣＭ１によって提供されるバックアップ依頼手段６１２０の代わりに、コントローラ２のプロセッサ７１１で実行されているタスクＣＭ２によって提供されるバックアップ依頼手段６２２０が記憶手段６２６０に格納された実行分担表と外部受付順位表を用いてバックアップを依頼することもできる。このために、実行分担表とタスク負荷表と外部受付順位表は全てのコントローラ、あるいは、全てのプロセッサの状態を記憶している。図５１の例示ではダウンを検出したコントローラのバックアップ依頼手段が作動するものとした。

【０１７８】本実施例はコントローラが４台のシステムであるが、本発明の主旨を逸脱しない範囲で２台以上のコントローラを持つシステムに拡張できる。さらに、本実施例では１台のコントローラが４個のプロセッサを含むものとしたが本発明の主旨を逸脱しない範囲で１個以上のプロセッサを持つシステムに拡張できる。

【０１７９】

【発明の効果】一つのコントローラを数値演算機能はもとよりニューロ，ファジィ，行列等を高速に演算するための並列演算機能を実行するプロセッサ，シーケンス制御演算を高速に実行するするプロセッサ、さらには、ネットワークとの通信機能を実行するプロセッサ等の複数のプロセッサから構成し、各機能を並列に実行可能とすることにより、幅広い分野に適用可能なコントローラが実現できる。このコントローラをネットワークワーク接続することにより、幅広い分野に適用可能分散制御システムが実現できる。

【０１８０】また、入出力及びプロセッサを機能プログラムできるチップを実現する方法も提供した。このことにより、多種多様な制御対象に対しても少ない種類のコントローラで対応でき、保守性の点で優れたシステムを構成できる。

【０１８１】各プロセッサをより小さい単位のプロセッ

サエレメントで構成することにより、フレキシビリティが向上しＬＳＩ化も容易になり、コントローラの小型化を図れる。従って、コントローラ本体の小型化が図れ、分散設置可能なコントローラが実現できる。また予備プロセッサ及び不揮発性メモリの予備領域を設けたことにより、部分故障に対してコントローラ自身で自己修復が可能となる。

【０１８２】また、バスの拡張が容易なバスコネクタつきの外観構造を採ることで小規模から大規模まで広い範囲の制御対象への対応が可能となる。

【０１８３】また、ユーザの書いたプログラムとニューラルネットワーク等の複数の出力生成手段によって得られる複数の出力を、学習状況や経時変化等に対し適応的に複合させてより良い制御を可能にすると共に、簡単な操作でユーザの好みの制御を行わせることができる。

【０１８４】さらに、本発明により分散制御システムにおけるコモンモード故障時に異常情報が多数ネットワークに発生し通信ネックが発生することを防止することができる。また、分散制御システムにおける故障箇所の同定が可能である。

【０１８５】さらにまた、本発明により分散制御システムのコントローラの多重ダウンに対しても負荷を自律的に稼働中のコントローラに分散しバックアップすることが可能である。さらにまた、本発明により分散制御システムのコントローラを複数のプロセッサで構成したため、同一コントローラ内のプロセッサの多重ダウンに対しても負荷を自律的に稼働中のプロセッサに分散しバックアップすることが可能である。このため、信頼性及び耐故障性を高めることが可能である。

【図面の簡単な説明】

【図１】本発明の一実施例であるところのプラント制御システム。

【図２】従来のの鉄鋼圧延制御システム。

【図３】具体的な本発明の応用例であるところの鉄鋼圧延制御システム。

【図４】別の具体的な本発明の応用例であるところの自動車の制御システム。

【図５】別の具体的な本発明の応用例であるところの自動車の制御システム。

【図６】本発明を階層システムに適用した場合のシステム構成例。

【図７】本発明の一実施例であるところのコントローラの内部構成図。

【図８】本発明の一実施例であるところのコントローラの内部構成図。

【図９】本発明の他の一実施例であるところのコントローラの内部構成図。

【図１０】本発明の他の一実施例であるところのコントローラの内部構成図。

【図１１】本発明のさらに他の一実施例であるところのコントローラの内部構成図。

【図１２】本発明のさらに他の一実施例であるところのコントローラの内部構成図。

【図１３】機能プログラム可能なコントローラの全体ブロック図。

【図１４】入出力間の接続を不揮発性メモリを用いてプログラマブルとしたスイッチアレイの構成例。

【図１５】図１０に示したスイッチアレイの中のメモリセルの構成例。

【図１６】スイッチアレイに予備領域を設けた実施例。

【図１７】マイクロプログラムメモリに不揮発性メモリを用いたプロセッサの内部構成例。

【図１８】各プロセッサをプロセッサエレメントの組み合せで構成した実施例。

【図１９】Ｉ／Ｏ８３の内部構成図。

【図２０】Ｉ／Ｏ８３を１チップの半導体で実現した構成図。

【図２１】Ｉ／Ｏ８３を１チップの半導体で実現した構成図。

【図２２】本発明のコントローラ１′を１チップの半導体で実現した構成図。

【図２３】本発明のコントローラ１′を１チップの半導体で実現した構成図。

【図２４】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図２５】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図２６】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図２７】本発明の一実施例であるところのコントローラの外観をそれぞれ示す図。

【図２８】本発明における制御演算部の基本構成図。

【図２９】本発明における制御演算部の全体概要図。

【図３０】出力計算部の構成。

【図３１】評価部の構成。

【図３２】複数の評価基準を有する場合の全体概要図。

【図３３】ニューラルネットによって重み値を生成する場合の基本構成図。

【図３４】診断機能つきの分散型コントローラ構成図。

【図３５】分散システムの各コントローラの制約条件の例図。

【図３６】各コントローラの通信インタフェースの構成図。

【図３７】異常情報フォーマット図。

【図３８】制約未充足時の異常信号送信処理の処理手順図。

【図３９】不要情報判定アルゴリズムの処理手順図。

【図４０】診断情報取込み判断処理手順図。

【図４１】故障情報の加工処理手順図。

【図４２】異常情報の包含関係を示す例図。

51

【図４３】自律分散制御システムの論理的構成図。
【図４４】自律的なバックアップ依頼手順図。
【図４５】自律的なバックアップ受付手順図。
【図４６】内部バックアップ手順図。
【図４７】自律分散制御システムの物理的構成図。
【図４８】実行分担表を示す例図。
【図４９】タスク負荷表を示す例図。
【図５０】外部受付順位表を示す例図。
【図５１】自律分散制御システムのバックアップ方法を示す例図。
【図５２】１回目の内部バックアップにおける受付用タスクキューの例図。
【図５３】２回目の内部バックアップにおける受付用タスクキューの例図。
【図５４】１回目の外部バックアップにおける受付用タスクキューの例図。
【図５５】内部バックアップ後の実行分担表を示す例図。
【図５６】内部バックアップ後の受付順位表を示す例図。

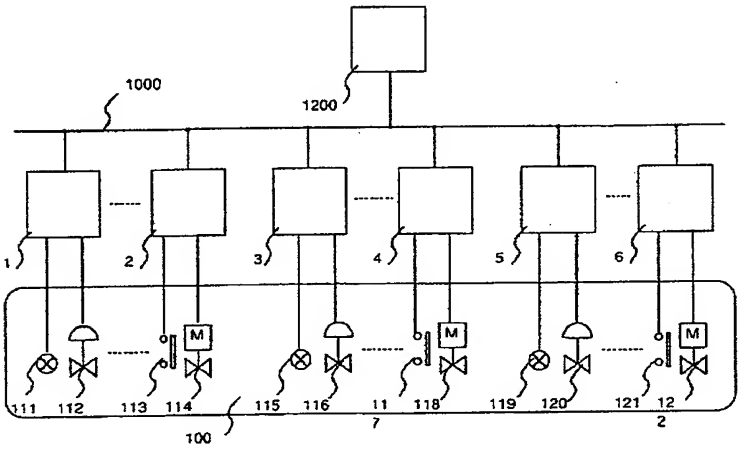
【符号の説明】

１，２，３，４，５，６，６０００…コントローラ、４０…ＮＣＰ、４１ａ，４１ｂ，４１ｃ，４１ｄ…ネットワーク接続用コネクタ、５２，８２，８２ａ，８２ｂ，８２ｃ，８２ｄ…グローバルメモリ、５３…Ｉ／Ｏ、５４，５６，５８，８４，８７，９０，９３…プロセッサ、５５，５７，５９，８５，８８，９１，９４…ローカルメモリ、６０，６１，６２，６３…バス拡張用コネクタ、６４…Ｉ／Ｏコネクタ、７０ａ，７０ｂ，７０ｃ，７０ｄ…バス、７１ａ，７１ｂ，７１ｃ，７１ｄ…シリアルデータ通信用のバス、７２ａ，７２ｂ，７２ｃ，72ｄ…故障通知用のバス、８３ａ，８３ｂ，８３ｃ，８３ｄ…入出力処理装置（Ｉ／Ｏ）、８６，８９，＊

52

＊９２，９５…ランアダプタ、１００…プラント、７０１，７０２，７０３，７０４…コントローラ１のプロセッサ、７１１，７１２，７１３，７１４…コントローラ２のプロセッサ、７２１，７２２，７２３，７２４…コントローラ３のプロセッサ、７３１，７３２，７３３，７３４…コントローラ４のプロセッサ、１０００…ネットワークワーク、２００１…センサ等からの入力データ、２００２…入力データ、２００３…出力データ、２００４…アクチュエータ等への出力データ、２０１０…センサ等、２０１１…入力処理部、２０１２…出力計算部、２０１３…出力処理部、２０１４…アクチュエータ等、２０１５…評価部、１２００…マンマシン装置、２０１７…タッチパネル，マウス等のポインティングデバイス、２０２１…重み付け部、２０２２…出力データ合成部、２１０１…ニューラルネットワーク、２１０２…出力生成手段２、２１０３…出力生成手段３、２１２２…重み値更新信号、６１１０，６２１０，６３１０，６４１０…通信手段、６１２０，６２２０，６３２０，６４２０…バックアップ依頼手段、６１３０，６２３０，６３３０，６４３０…バックアップ受付手段、６１４０，６２４０，６３４０，６４４０…スケジューラ、６１５０，６２５０，６３５０，６４５０…制御手段、６１６０，６２６０，６３６０，６４６０…記憶手段、６１７０，６２７０，６３７０，６４７０…内部バックアップ手段、６１８０，６２８０，６３８０，６４８０…切り離し手段、６８１１…汎用エンジン（ＧＥ）、６８１２…ニューラルエンジン（ＮＥ）、６８１３…シーケンスエンジン（ＳＥ）、６８１４…ランアダプタ（ＬＡ）、６８１５…フリーランタイマ（ＦＲＴ）、６８１６…ローカルメモリ（ＬＭ）、６８１７…バスインタフェース（ＢＩＦ）、６８１８…ダイレクトメモリアクセス制御（ＤＭＡＣ）、６８１９…プロセッサ内のバス、６８２０…故障制御回路。

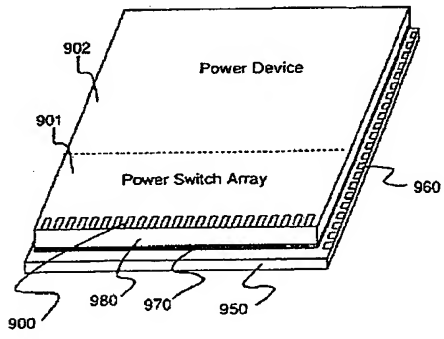
【図１】



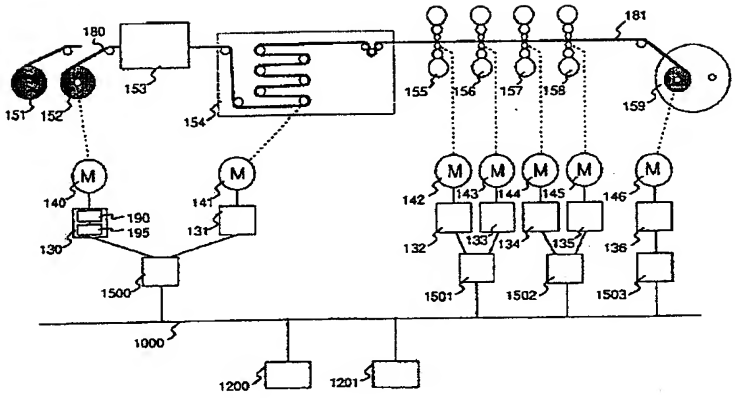
図一

【図２０】

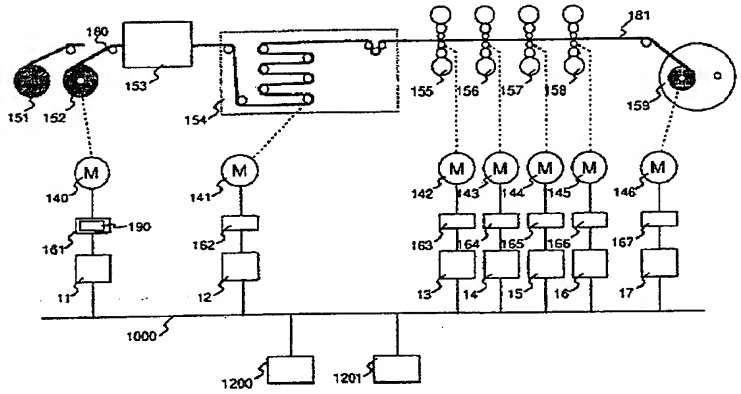
図 20



【図２】



【図３】



【図４】

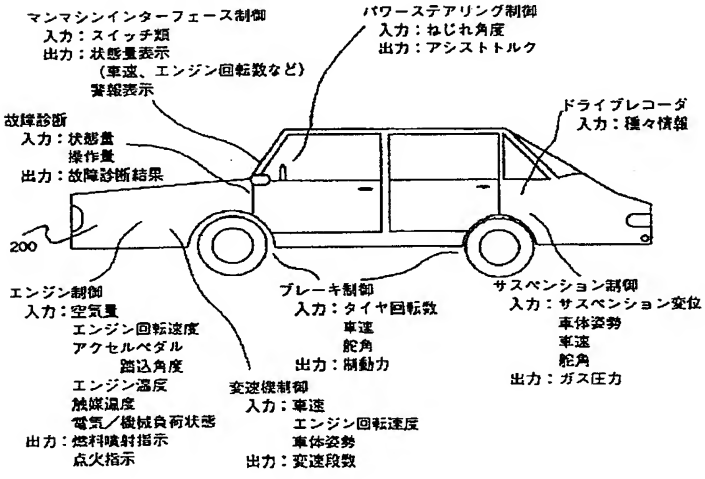


図４

【図２１】

図 21

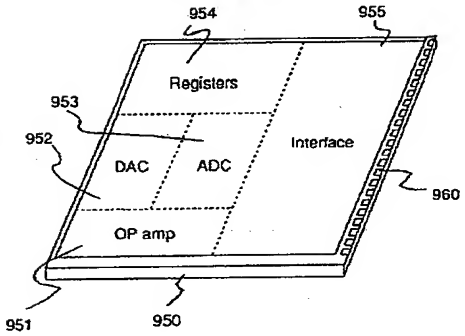


図２

【図２２】

図 22

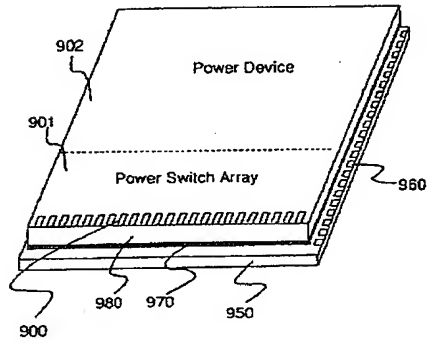


図３

【図５】

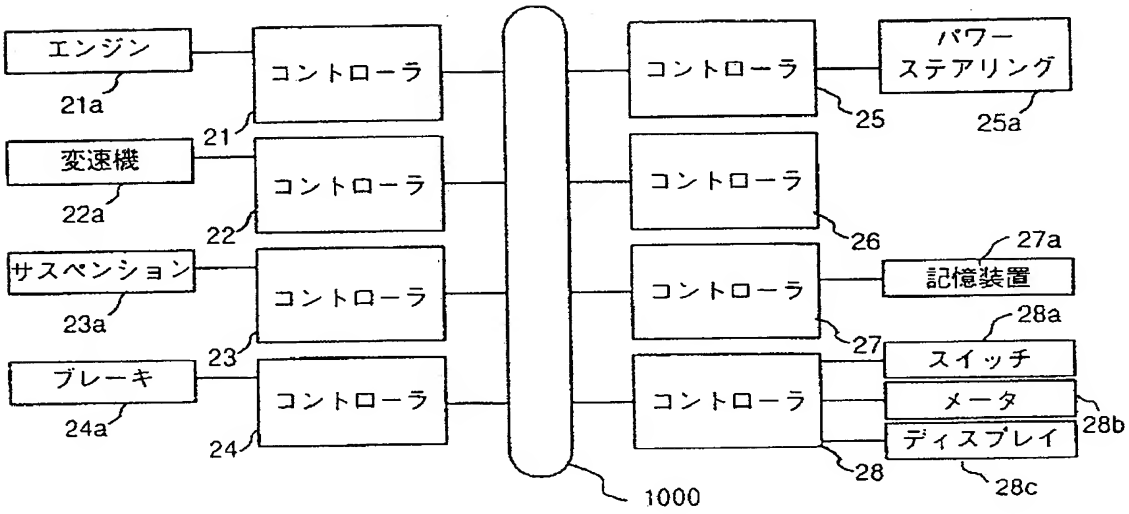


図５

【図６】

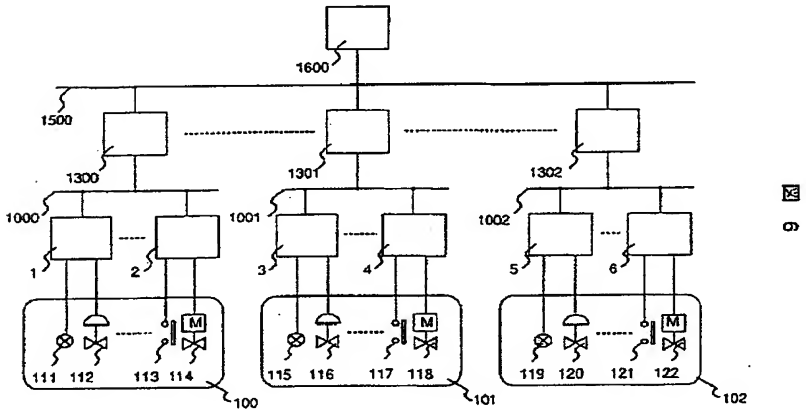
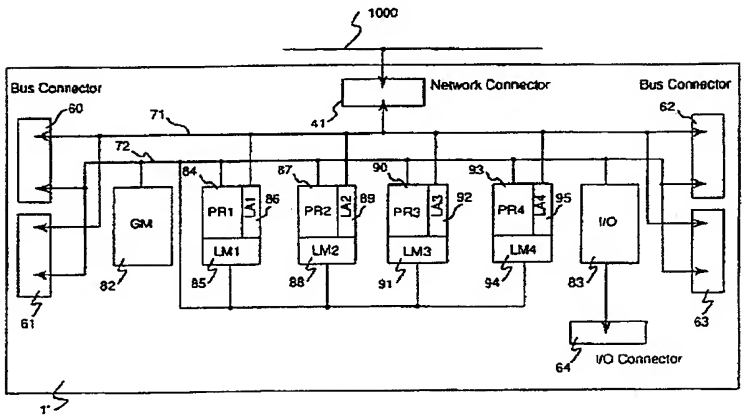


図６

【図９】



【図２３】

図 23

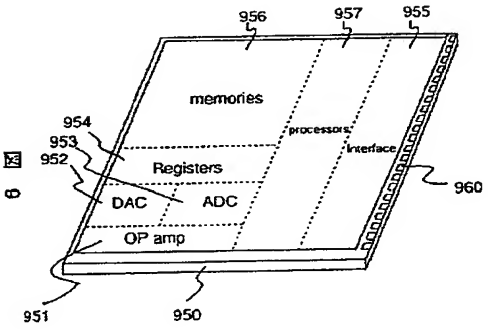


図９

【図７】

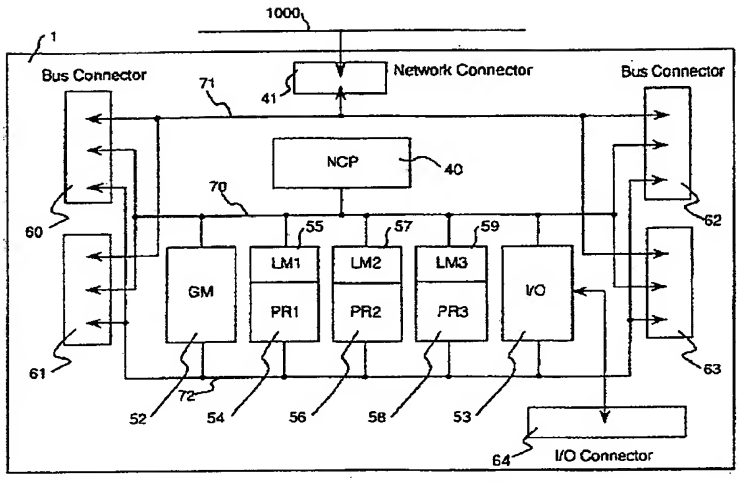


図 7

【図５０】

図　50

外部受付順位表	
順位	ﾏｲｸﾛｺﾝﾋﾟｭｰﾀ
1	DC1.PR4
2	DC2.PR2
3	DC3.PR1
4	DC2.PR3
5	DC3.PR3
6	DC2.PR1
・	・
・	以下省略 ・
・	・
・	・

【図８】

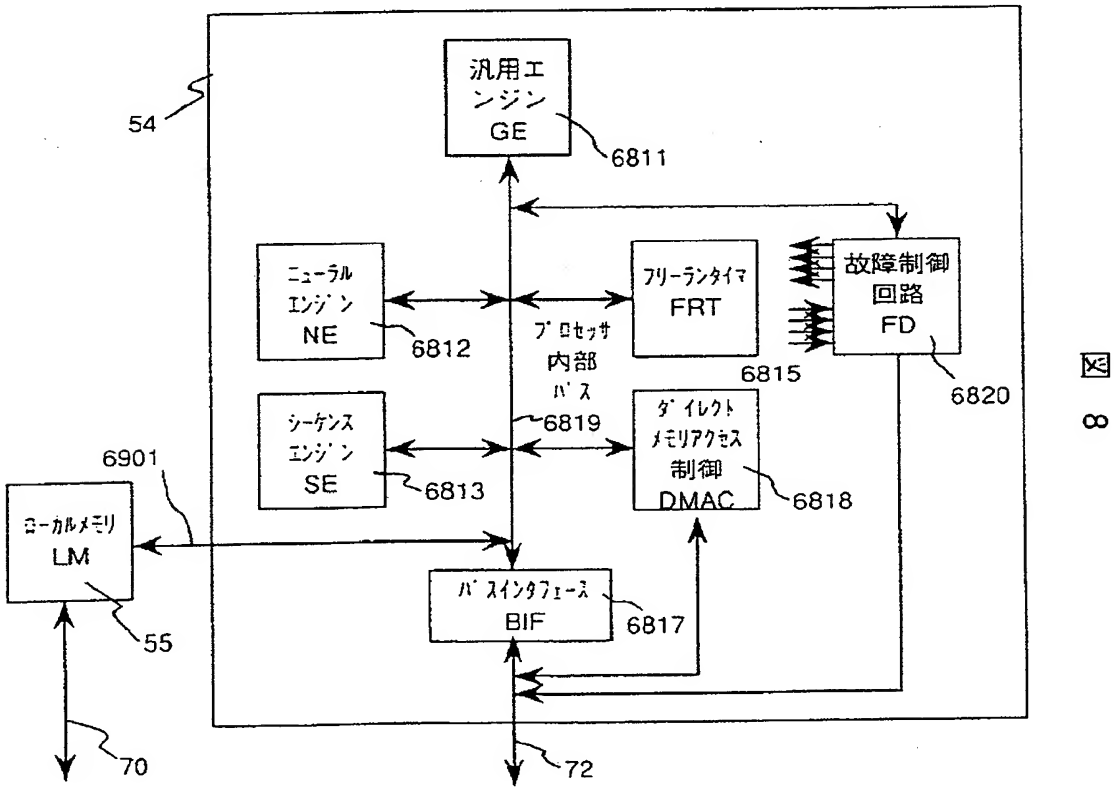


図 8

【図１０】

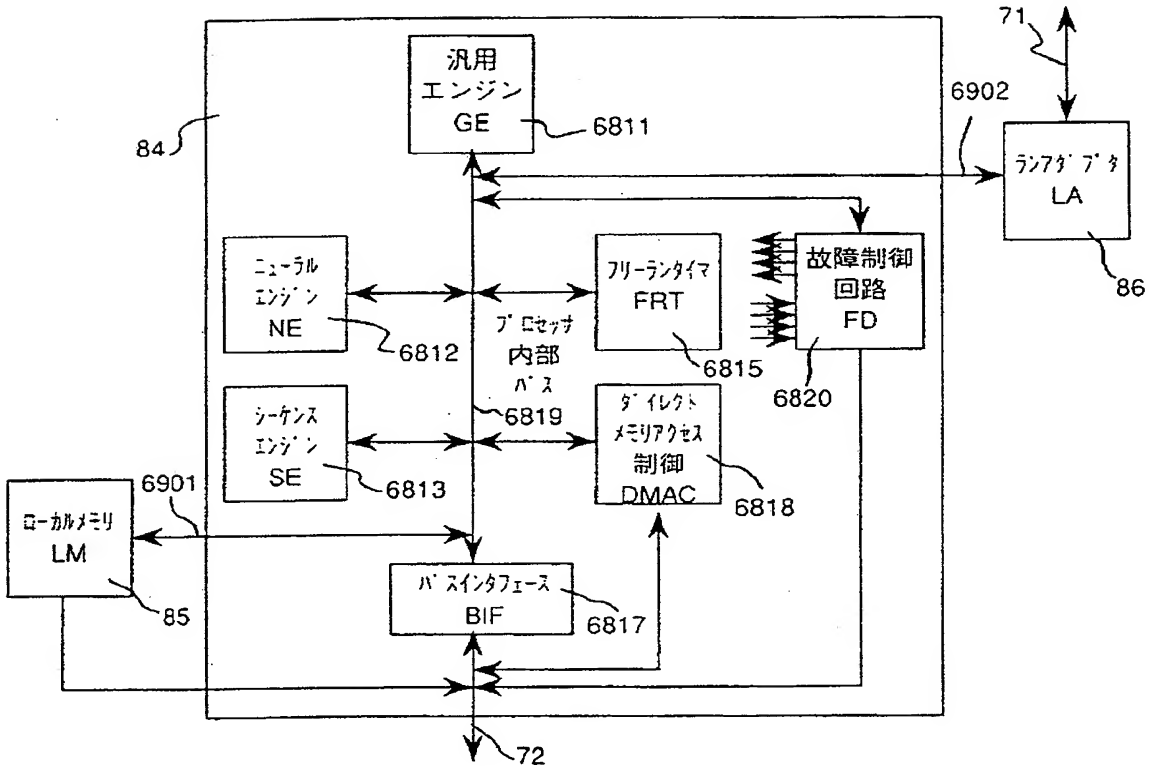
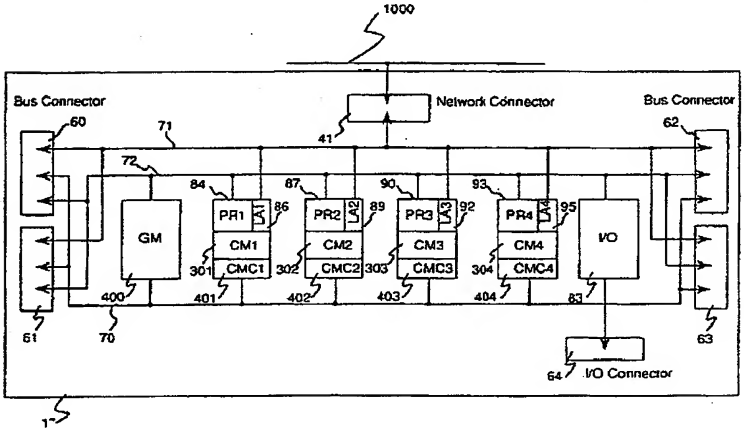


図 10

【図１１】



【図２５】

図 25

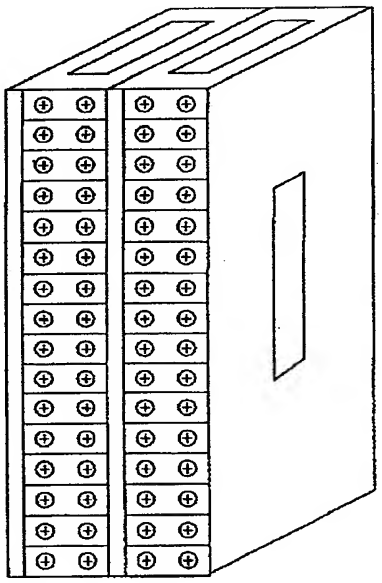


図 11

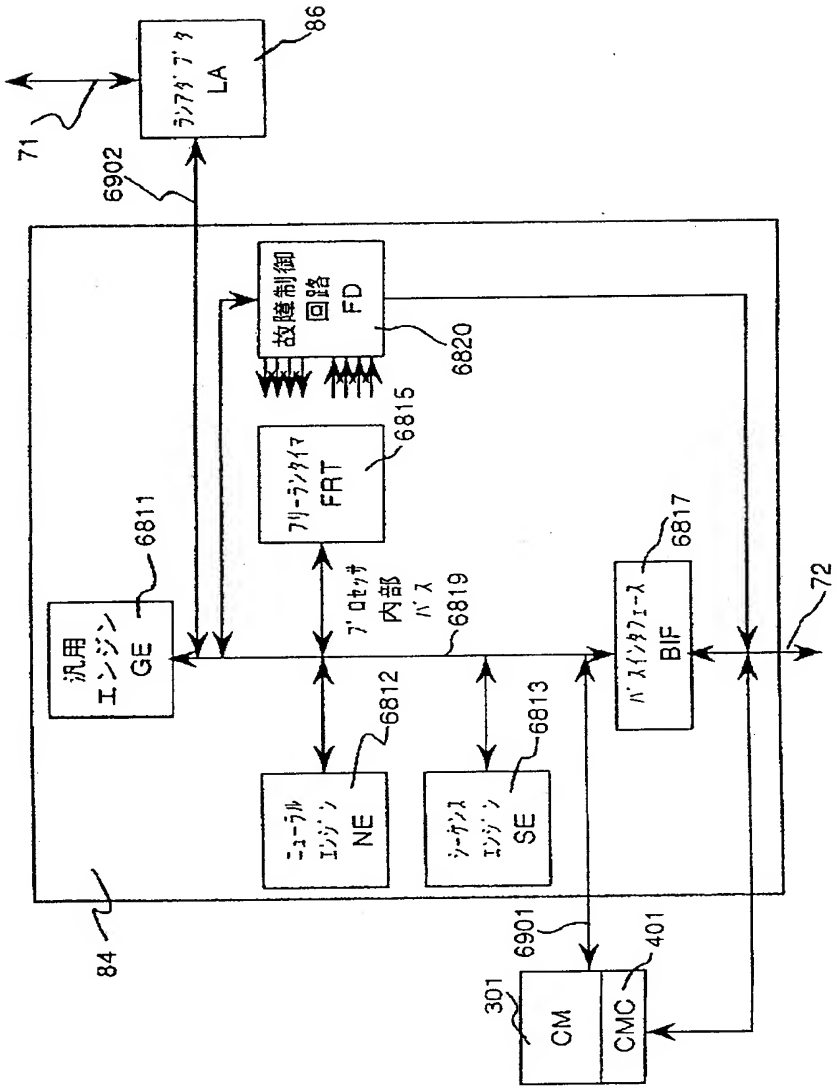
【図５４】

図 54

受付用タスクキュー				
タスク	負荷率（%）	属性	原籍	選択結果
OS22	5	fixed	DC2.PR2	選択
T21	18	communicative	DC2	選択
T18	26	somewhere	—	選択
T22	35	somewhere	—	選択

【図１２】

図 12


86
LA
71
6902
故障制御回路 FD
6820
6815
FRT
6811
汎用エンジン GE
6819
6817
BIF
72
6812
NE
6813
SE
84
6901
401
301
CM
CMC

【図１３】

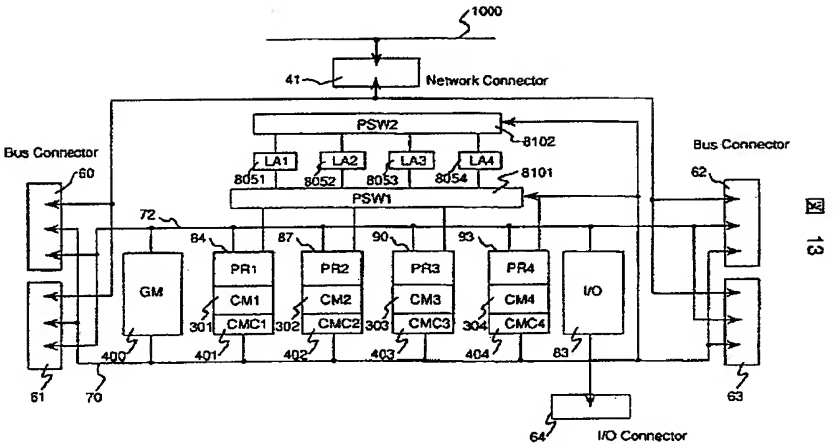
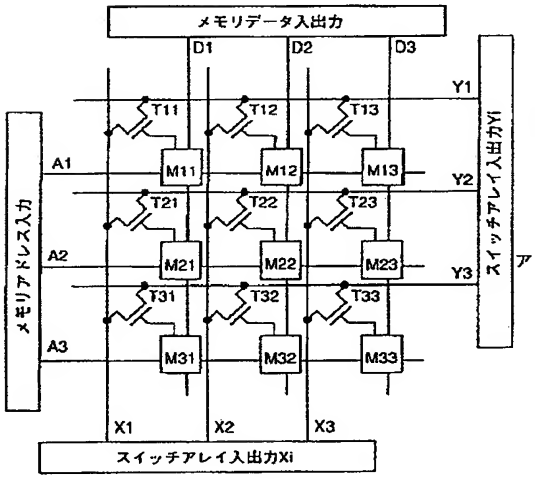


図 13

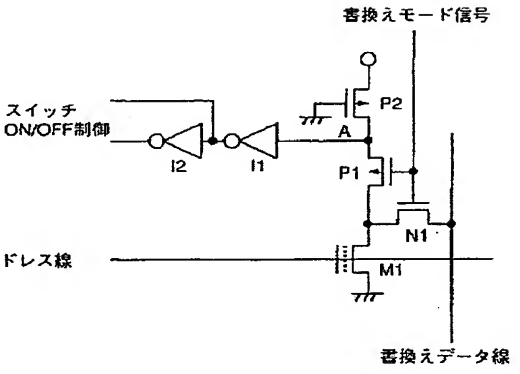
【図１４】

図 14



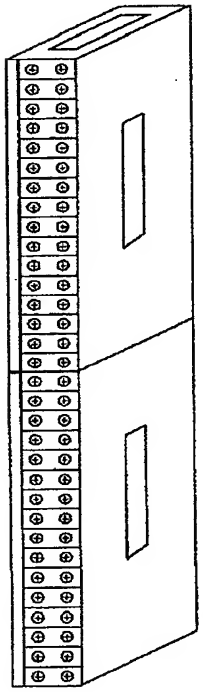
【図１５】

図 15



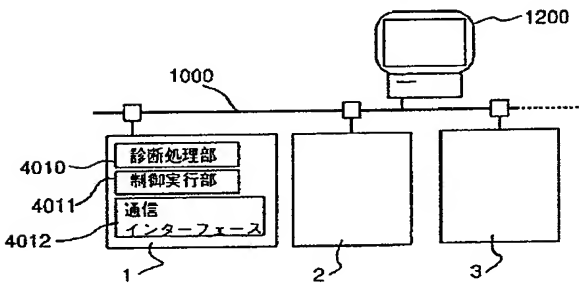
【図２６】

図 26



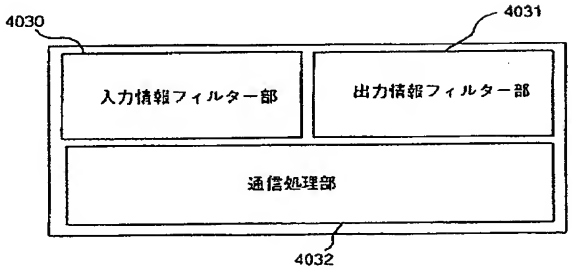
【図３４】

図 34



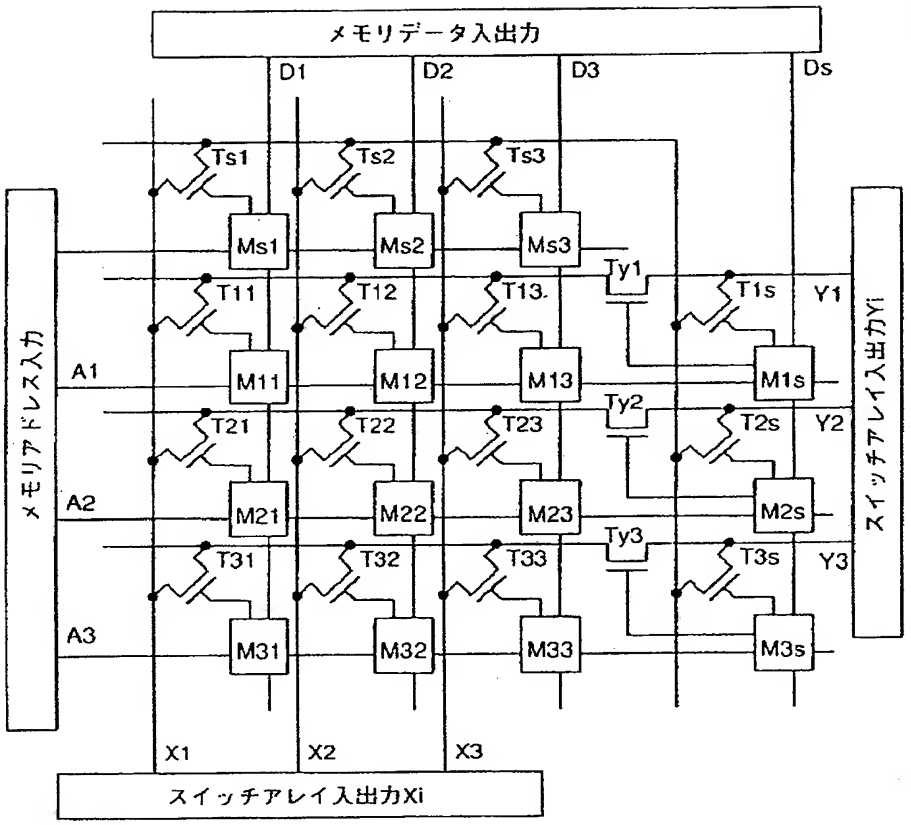
【図３６】

図 36



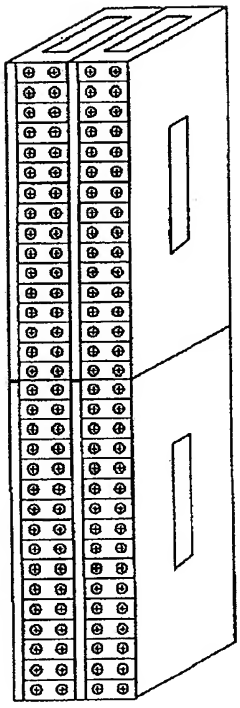
【図１６】

図　16



【図２７】

図　27



【図１９】

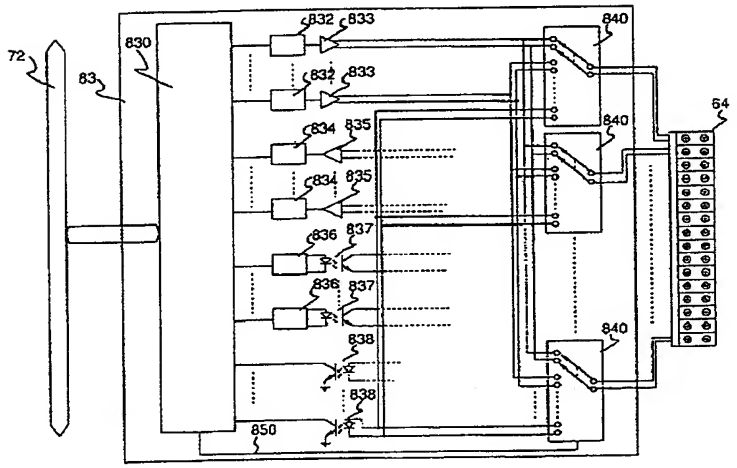


図　19

【図５６】

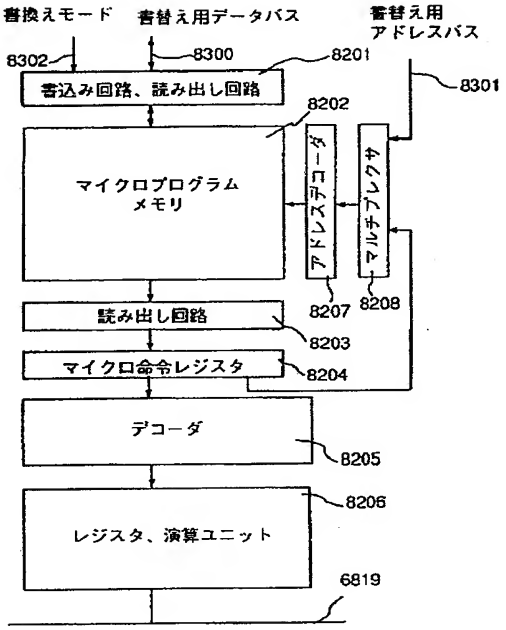
図　56

外部受付順位表	
順位	マイクロコンピュータ
1	DC2.PR2
2	DC3.PR1
3	DC2.PR3
4	DC3.PR3
5	DC2.PR1
6	DC3.PR2
・	以下省略
・	・
・	・

左記の様に網掛けした部分が更新されている

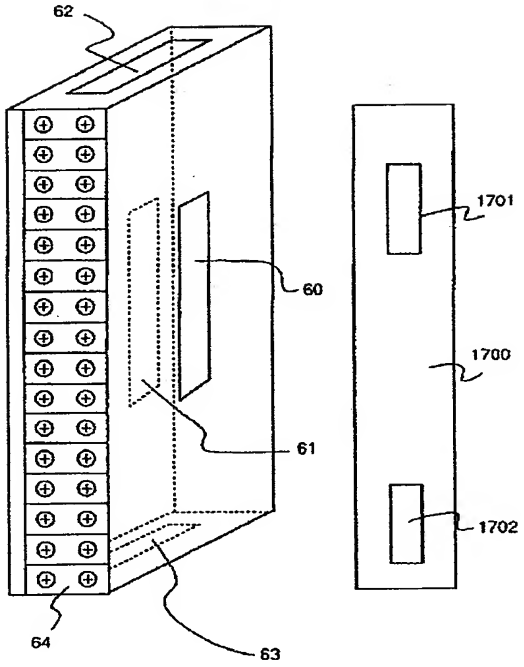
【図１７】

図 17



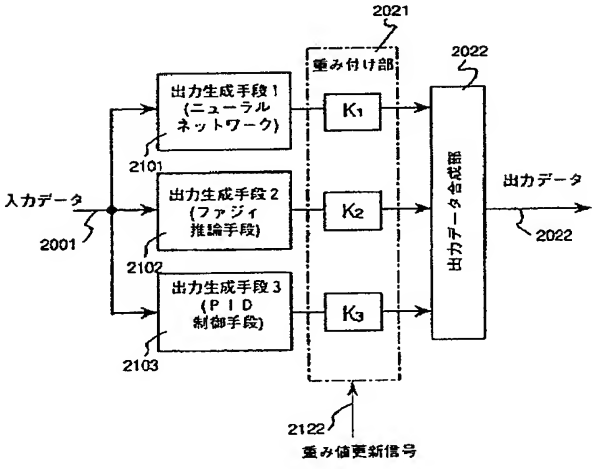
【図２４】

図 24



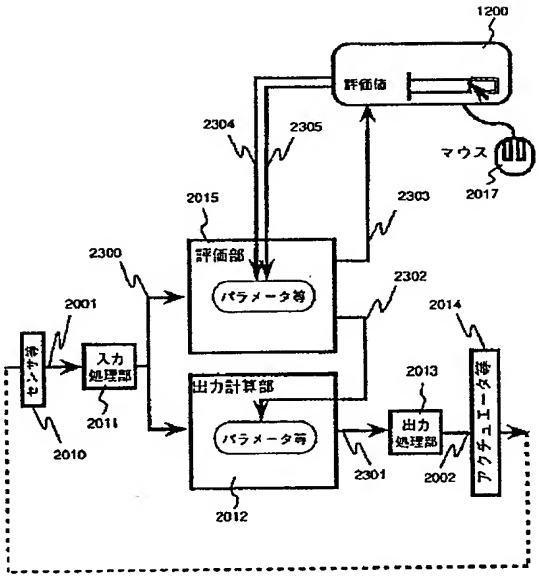
【図２８】

図 28



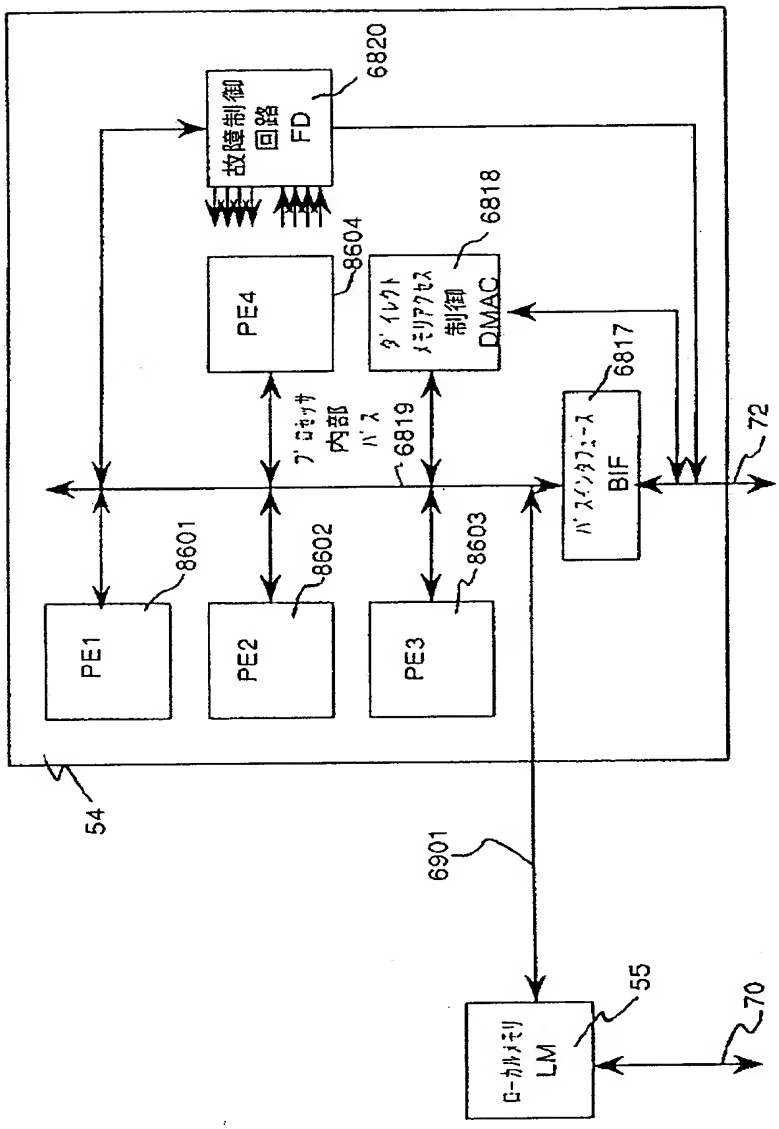
【図２９】

図 29



【図１８】

図　18


故障制御回路 FD
6820
PE4
8604
ダイレクトメモリアクセス制御 DMAC
6818
プロセッサ内部バス
6819
バスインタフェース BIF
6817
72
PE1
8601
PE2
8602
PE3
8603
54
6901
ローカルメモリ LM
55
70

【図３０】

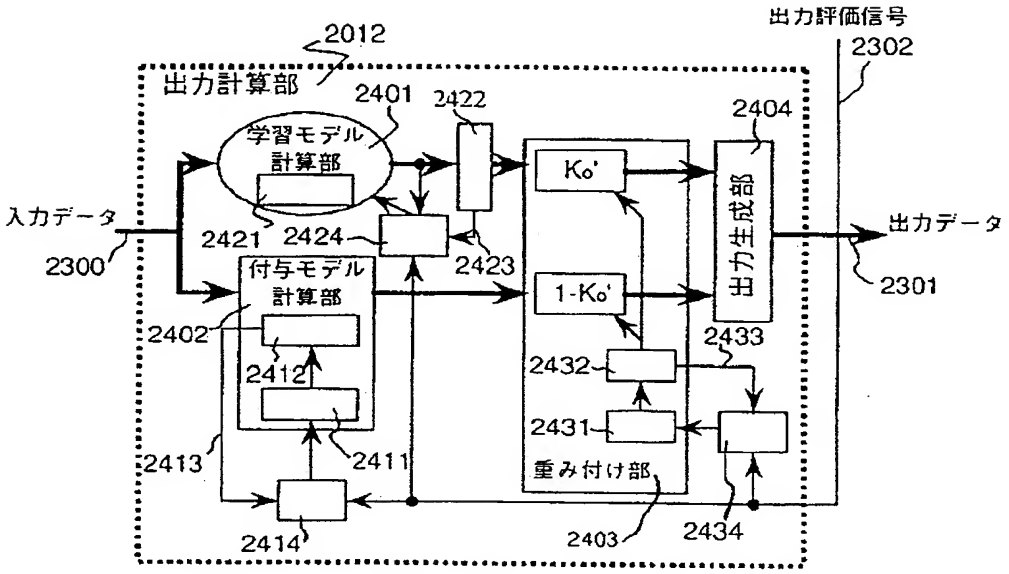


図 30

【図３１】

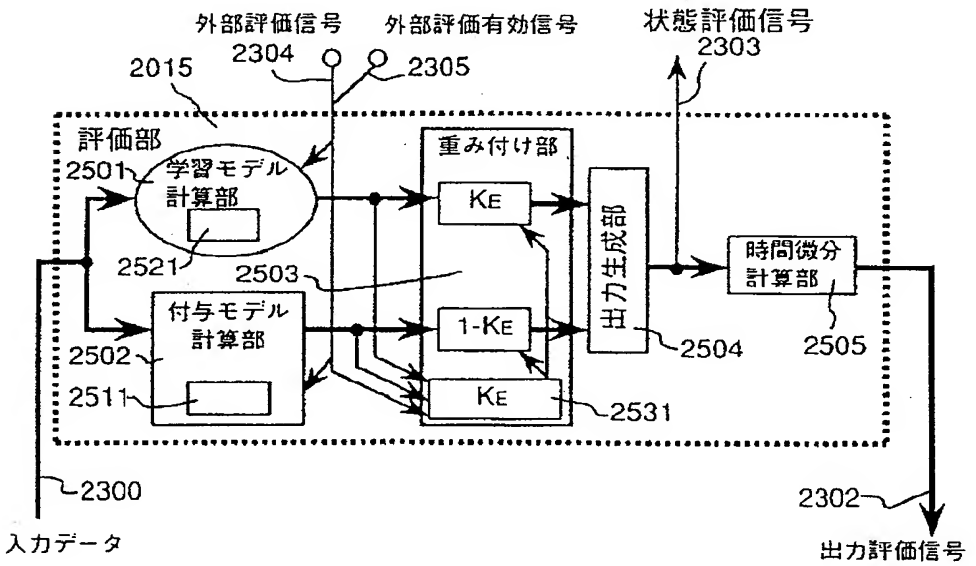


図 31

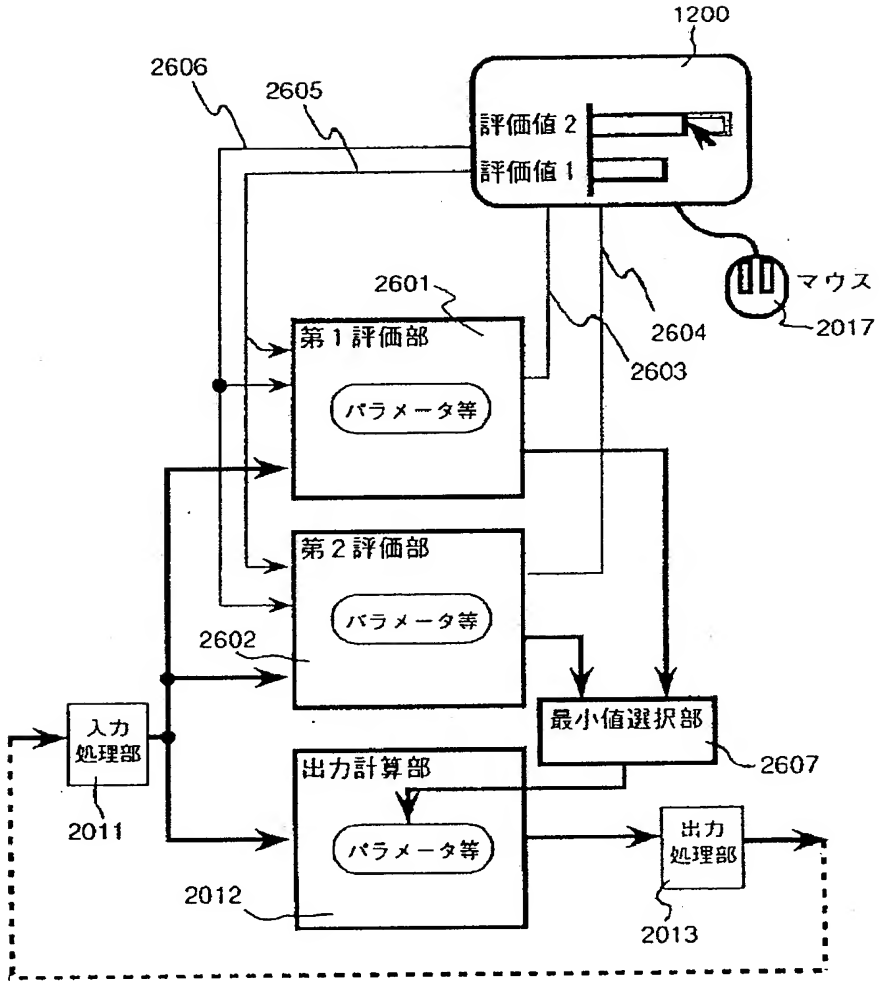
【図５２】

図 52

受付用タスクキュー				
タスク	負荷率（%）	属性	原籍	選択結果
OS14	5	fixed	DC1.PR4	選択
NC1	26	internally	DC1	選択
CM1	28	internally	DC1	選択
T11	13	communicative	DC1	非選択
T17	25	communicative	DC1	選択
T18	26	somewhere	—	追い出し

【図３２】

図　32



【図３７】

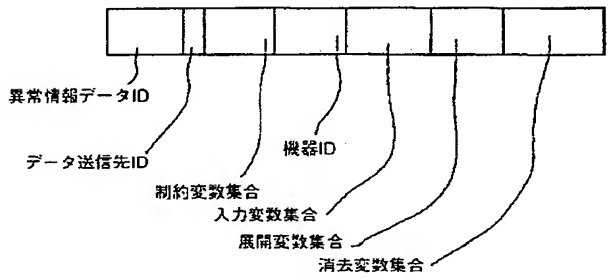


図 37

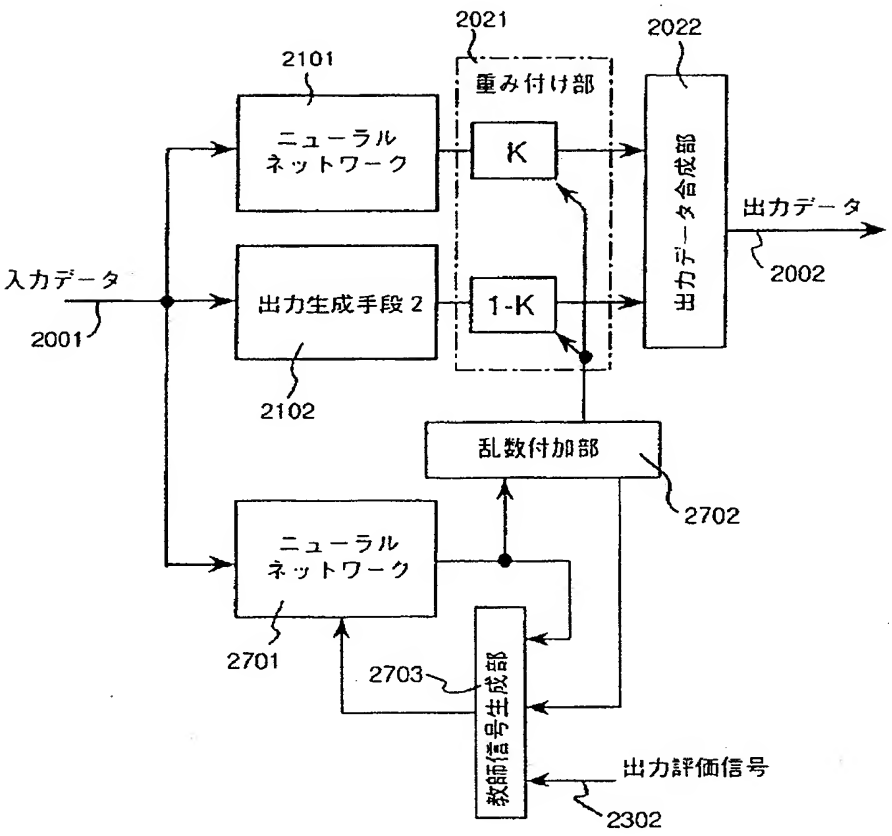
【図４８】

図　48

実行分担表			
ﾏｲｸﾛｺﾝﾋﾟｭｰﾀ	タスク	負荷率（%）	内部負荷順位
DC1. PR1	OS11, NC1, CM1, T11	72	2
DC1. PR2	OS12, T12, T13, T14	81	4
DC1. PR3	OS13, T15, T16	75	3
DC1. PR4	OS14, T17, T18	57	1
DC2. PR1	OS21, NC2, CM2	63	3
DC2. PR2	OS22, T21, T22	58	1
DC2. PR3	OS23, T23, T24	60	2
DC2. PR4	OS24, T25, T26	75	4
DC3. PR1	OS31, NC3, CM3	58	1
DC3. PR2	OS32, T31, T32	72	3
DC3. PR3	OS33, T33, T34, T35	60	2
DC3. PR4	OS34, T36, T37	74	4
・	・	・	・
・	・ 以下省略	・	・

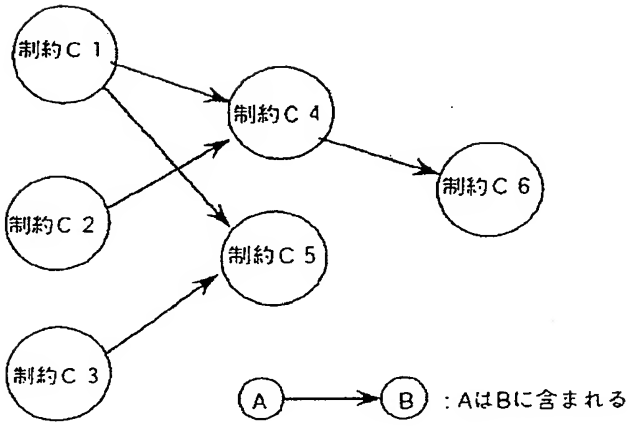
【図３３】

図　33



【図４２】

図　42



【図５３】

図　53

受付用タスクキュー				
タスク	負荷率（%）	属性	原籍	選択結果
OS13	5	fixed	DC1,PR3	選択
T11	13	communicative	DC1	選択
T16	41	communicative	DC1	選択
T18	26	somewhere	—	非選択
T15	28	somewhere	—	選択

【図３５】

図　35

	コントローラ1	コントローラ2	コントローラ3	コントローラ4
入力変数	X1,X2	X3,X4	X3,X5	X3,X6
出力変数	X3	X5,X6	X7	X8,X9
制約条件	C1: E(X3, X1+X2, e1)	C2: E(X5, X3+X4, e2) C3: E(X6, X3, e3) C4: X5 < 10	C5: E(X7, X3 * X5, e5) C6: X7 < 20	C7: E(X9, X3-X6+X5), e7) C8: E(X8, X3*X4, e8) C9: X8 < 5
制御関数	T1: X3 = X1+X2	T2: X5=X3+X4 T3: X6 = X3	T4: X7=X3*X5	T5: X9= X3-X6+ X5 T6: X8=X3*X4

【図５５】

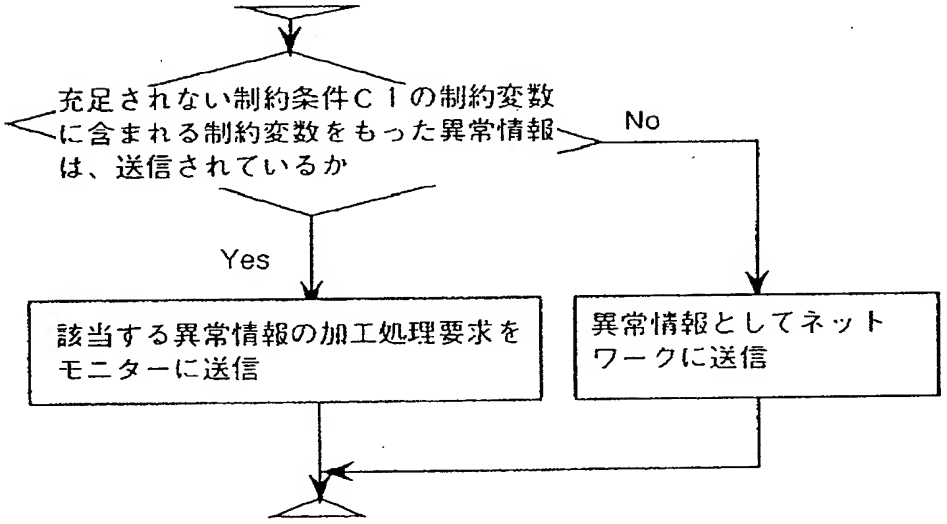
実行分担表			
ﾏｲｸﾛｺﾝﾋﾟｭｰﾀ	タスク	負荷率（%）	内部負荷順位
DC1.PR1	ﾀﾞｳﾝ	ﾀﾞｳﾝ	ﾀﾞｳﾝ
DC1.PR2	OS12 T12, T13, T14	81	1
DC1.PR3	OS13, T15, T16, T11	88	3
DC1.PR4	OS14, NC1, CM1, T17	84	2
DC2.PR1	OS21, NC2, CM2	63	3
DC2.PR2	OS22, T21, T22	58	1
DC2.PR3	OS23, T23, T24	60	2
DC2.PR4	OS24, T25, T26	75	4
DC3.PR1	OS31, NC3, CM3	58	1
DC3.PR2	OS32, T31, T32	72	3
DC3.PR3	OS33, T33, T34, T35	60	2
DC3.PR4	OS34, T36, T37	74	4
・	・以下省略	・	・
・	・	・	・

ﾀﾞｳﾝ　左記の様に網掛けした部分が更新されている

図 55

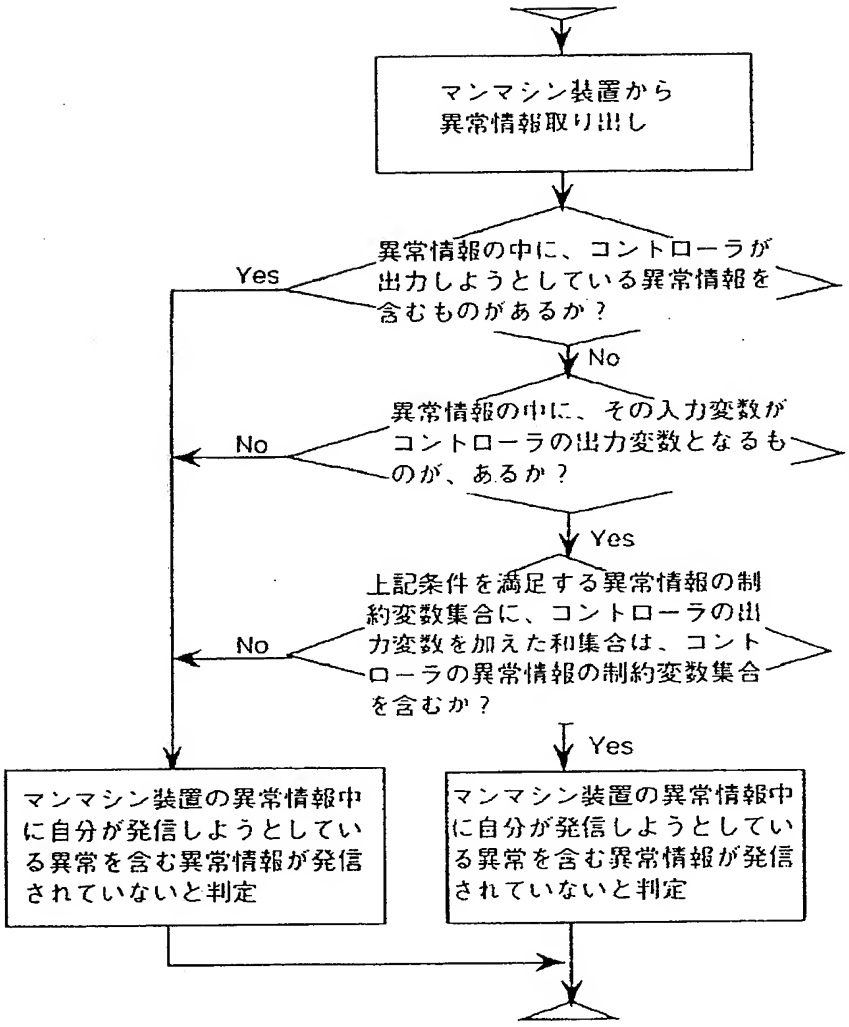
【図３８】

図　38



【図３９】

図　39



【図４９】

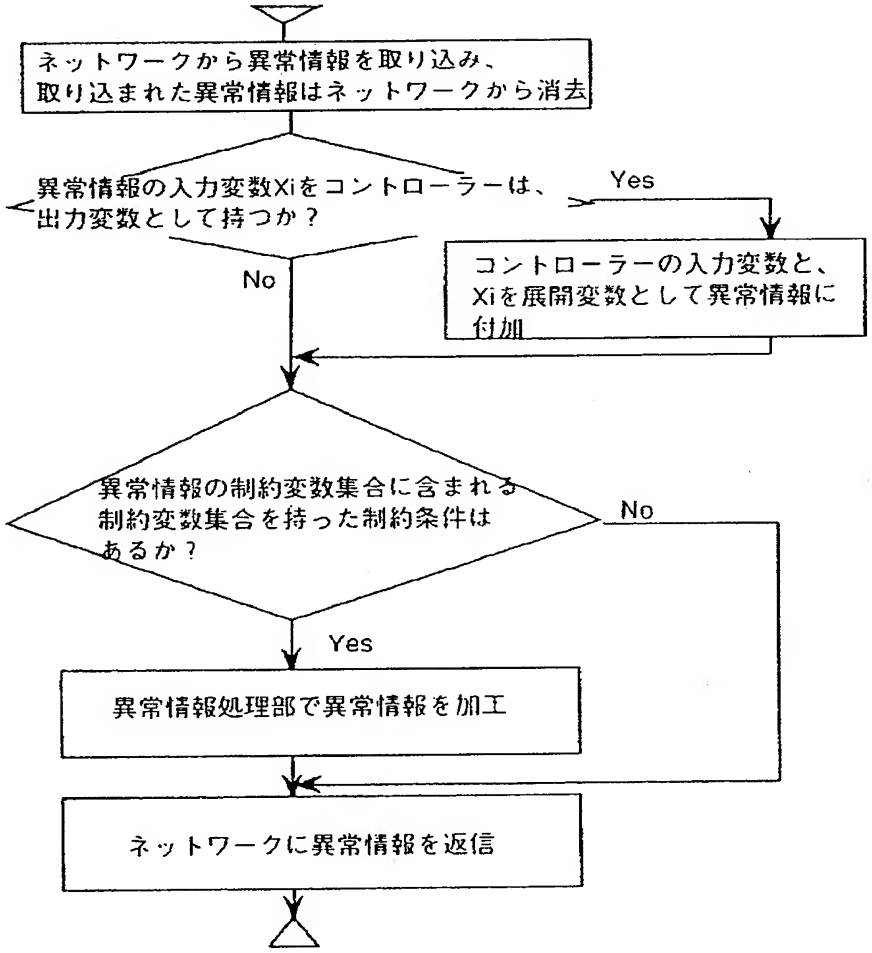
図　49

タスク負荷表

タスク	負荷率（%）	属性	原籍
NC1	26	internally	DC1.PR1
CM1	28	internally	DC1.PR1
OS11	5	fixed	DC1.PR1
OS12	5	fixed	DC1.PR2
OS13	5	fixed	DC1.PR3
OS14	5	fixed	DC1.PR4
T11	13	communicative	DC1.PR1
T12	24	somewhere	DC1.PR2
T13	31	communicative	DC1.PR2
T14	21	somewhere	DC1.PR2
T15	28	somewhere	DC1.PR3
T16	41	communicative	DC1.PR3
T17	25	communicative	DC1.PR4
T18	26	somewhere	DC1.PR4
NC2	27	internally	DC2.PR1
・	・	・	・
・	・ 中略	・	・
・	・	・	・
OS22	5	fixed	DC2.PR2
OS23	5	fixed	DC2.PR3
・	・ 中略	・	・
・	・	・	・
T21	18	communicative	DC2.PR2
T22	35	somewhere	DC2.PR2
T23	23	communicative	DC2.PR3
T24	32	somewhere	DC2.PR3
・	・ 中略	・	・
・	・	・	・
NC3	27	internally	DC3.PR1
CM3	26	internally	DC3.PR1
OS31	5	fixed	DC3.PR1
・	・ 以下省略	・	・
・	・	・	・

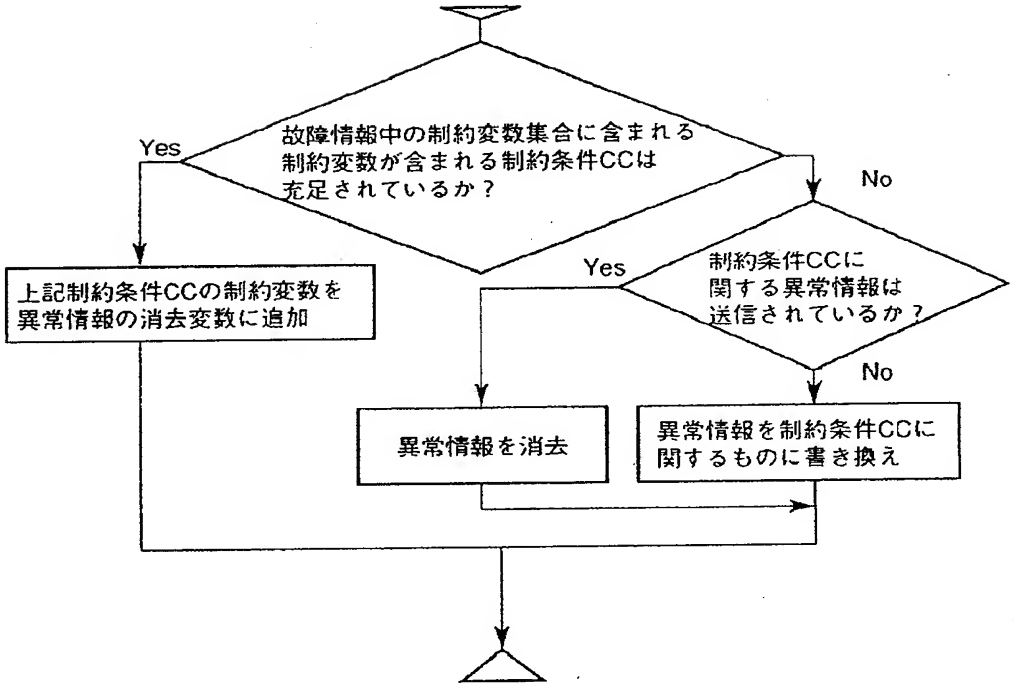
【図４０】

図　40


ネットワークから異常情報を取り込み、
取り込まれた異常情報はネットワークから消去
異常情報の入力変数Xiをコントローラーは、
出力変数として持つか？
Yes
No
コントローラーの入力変数と、
Xiを展開変数として異常情報に
付加
異常情報の制約変数集合に含まれる
制約変数集合を持った制約条件は
あるか？
No
Yes
異常情報処理部で異常情報を加工
ネットワークに異常情報を返信

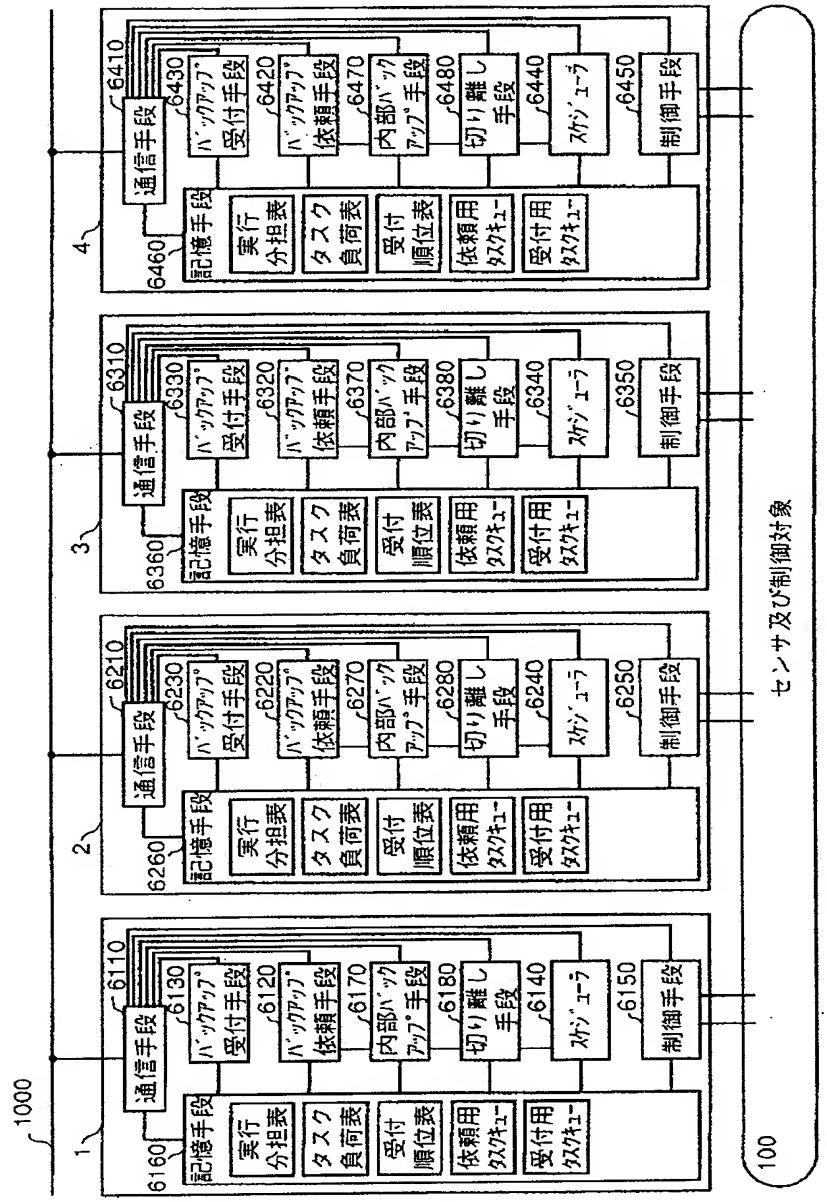
【図４１】

図　41


故障情報中の制約変数集合に含まれる
制約変数が含まれる制約条件CCは
充足されているか？
Yes
No
上記制約条件CCの制約変数を
異常情報の消去変数に追加
制約条件CCに
関する異常情報は
送信されているか？
Yes
No
異常情報を消去
異常情報を制約条件CCに
関するものに書き換え

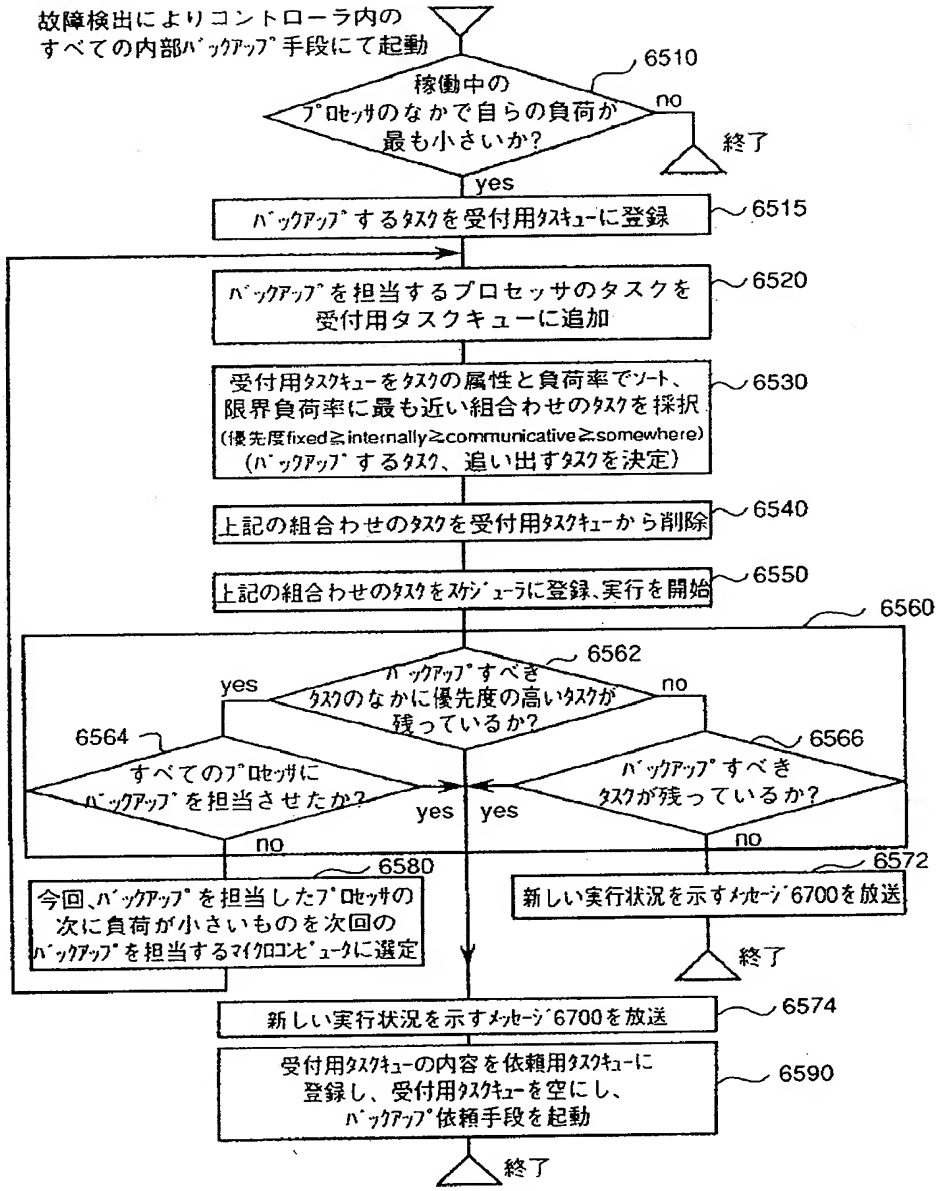
【図４３】

図　43


1000
1
2
3
4
通信手段
記憶手段
実行分担表
タスク負荷表
受付順位表
依頼用タスクキュー
受付用タスクキュー
バックアップ手段
受付手段
依頼手段
内部バックアップ手段
切り離し手段
スケジューラ
制御手段
センサ及び制御対象
100

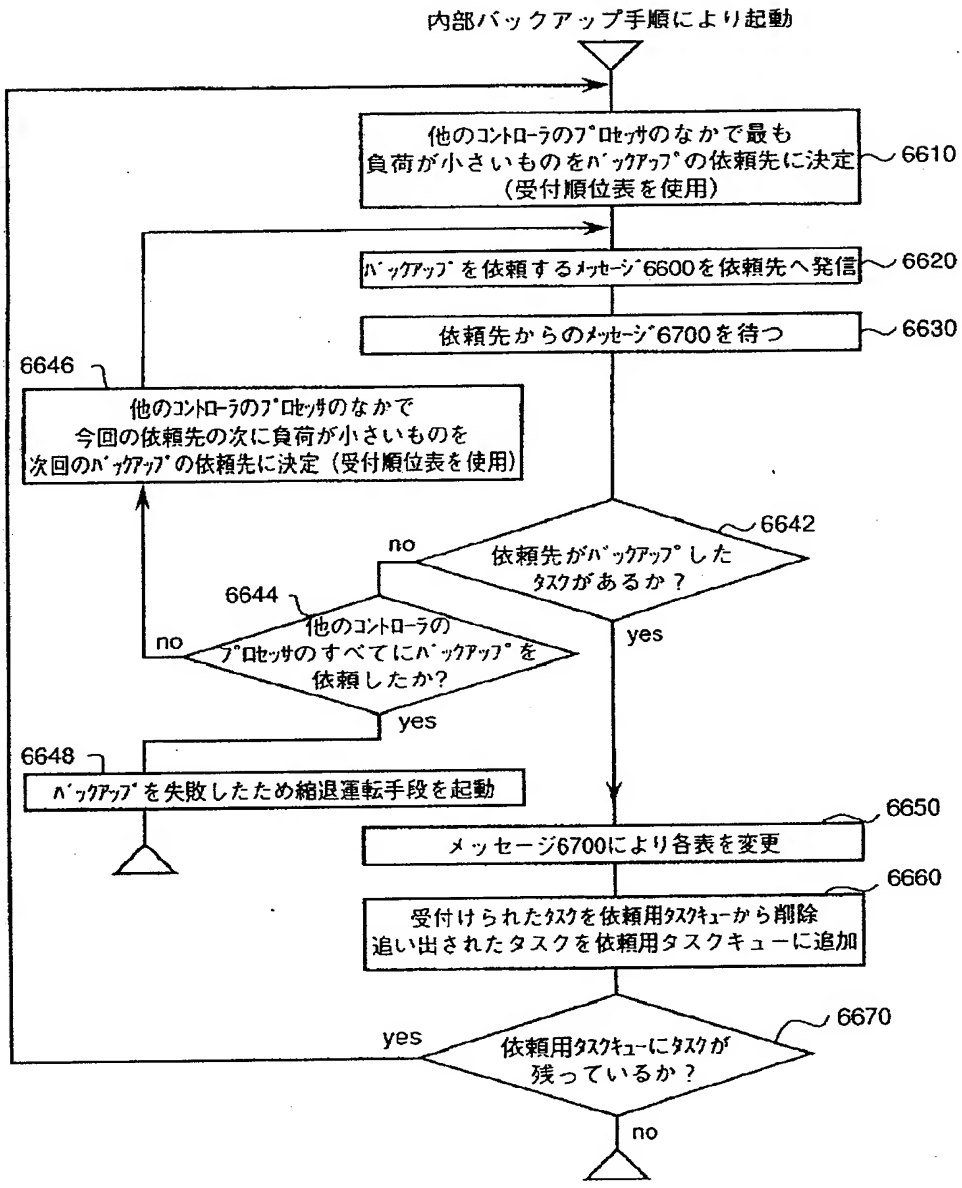
【図４４】

図　44


故障検出によりコントローラ内の
すべての内部バックアップ手段にて起動
6510
稼働中の
プロセッサのなかで自らの負荷が
最も小さいか?
no
終了
yes
バックアップするタスクを受付用タスキューに登録
6515
バックアップを担当するプロセッサのタスクを
受付用タスクキューに追加
6520
受付用タスクキューをタスクの属性と負荷率でソート、
限界負荷率に最も近い組合わせのタスクを採択
(優先度fixed≧internally≧communicative≧somewhere)
(バックアップするタスク、追い出すタスクを決定)
6530
上記の組合わせのタスクを受付用タスクキューから削除
6540
上記の組合わせのタスクをスケジューラに登録、実行を開始
6550
6560
6562
バックアップすべき
タスクのなかに優先度の高いタスクが
残っているか?
yes
no
6564
すべてのプロセッサに
バックアップを担当させたか?
6566
バックアップすべき
タスクが残っているか?
yes
yes
no
no
6580
今回、バックアップを担当したプロセッサの
次に負荷が小さいものを次回の
バックアップを担当するマイクロコンピュータに選定
6572
新しい実行状況を示すメッセージ6700を放送
終了
新しい実行状況を示すメッセージ6700を放送
6574
受付用タスクキューの内容を依頼用タスクキューに
登録し、受付用タスクキューを空にし、
バックアップ依頼手段を起動
6590
終了

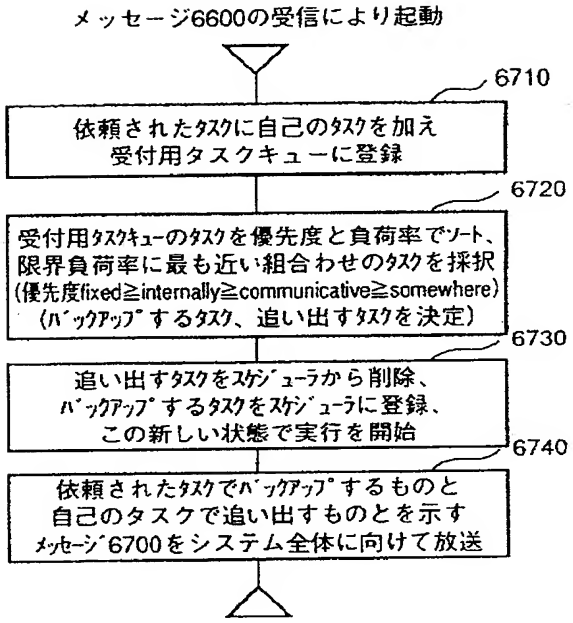
【図４５】

図　45


内部バックアップ手順により起動
他のコントローラのプロセッサのなかで最も負荷が小さいものをバックアップの依頼先に決定（受付順位表を使用）
6610
バックアップを依頼するメッセージ6600を依頼先へ発信
6620
依頼先からのメッセージ6700を待つ
6630
6646
他のコントローラのプロセッサのなかで今回の依頼先の次に負荷が小さいものを次回のバックアップの依頼先に決定（受付順位表を使用）
6642
依頼先がバックアップしたタスクがあるか？
no
yes
6644
他のコントローラのプロセッサのすべてにバックアップを依頼したか?
no
yes
6648
バックアップを失敗したため縮退運転手段を起動
6650
メッセージ6700により各表を変更
6660
受付けられたタスクを依頼用タスクキューから削除
追い出されたタスクを依頼用タスクキューに追加
6670
依頼用タスクキューにタスクが残っているか？
yes
no

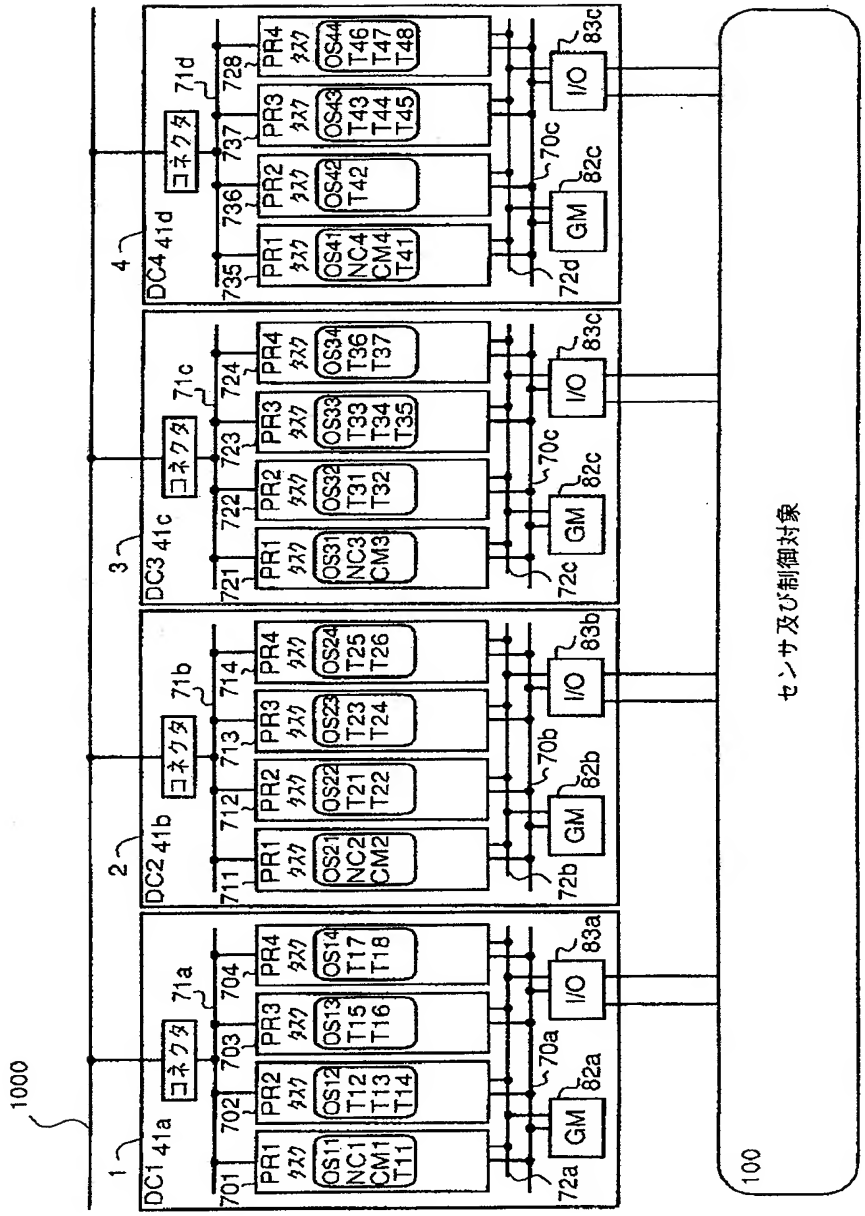
【図４６】

図 46


メッセージ6600の受信により起動
6710
依頼されたﾀｽｸに自己のﾀｽｸを加え
受付用タスクキューに登録
6720
受付用ﾀｽｸｷｭｰのﾀｽｸを優先度と負荷率でｿｰﾄ、
限界負荷率に最も近い組合わせのﾀｽｸを採択
(優先度fixed≧internally≧communicative≧somewhere)
(ﾊﾞｯｸｱｯﾌﾟするﾀｽｸ、追い出すﾀｽｸを決定)
6730
追い出すﾀｽｸをｽｹｼﾞｭｰﾗから削除、
ﾊﾞｯｸｱｯﾌﾟするﾀｽｸをｽｹｼﾞｭｰﾗに登録、
この新しい状態で実行を開始
6740
依頼されたﾀｽｸでﾊﾞｯｸｱｯﾌﾟするものと
自己のタスクで追い出すものとを示す
ﾒｯｾｰｼﾞ6700をシステム全体に向けて放送

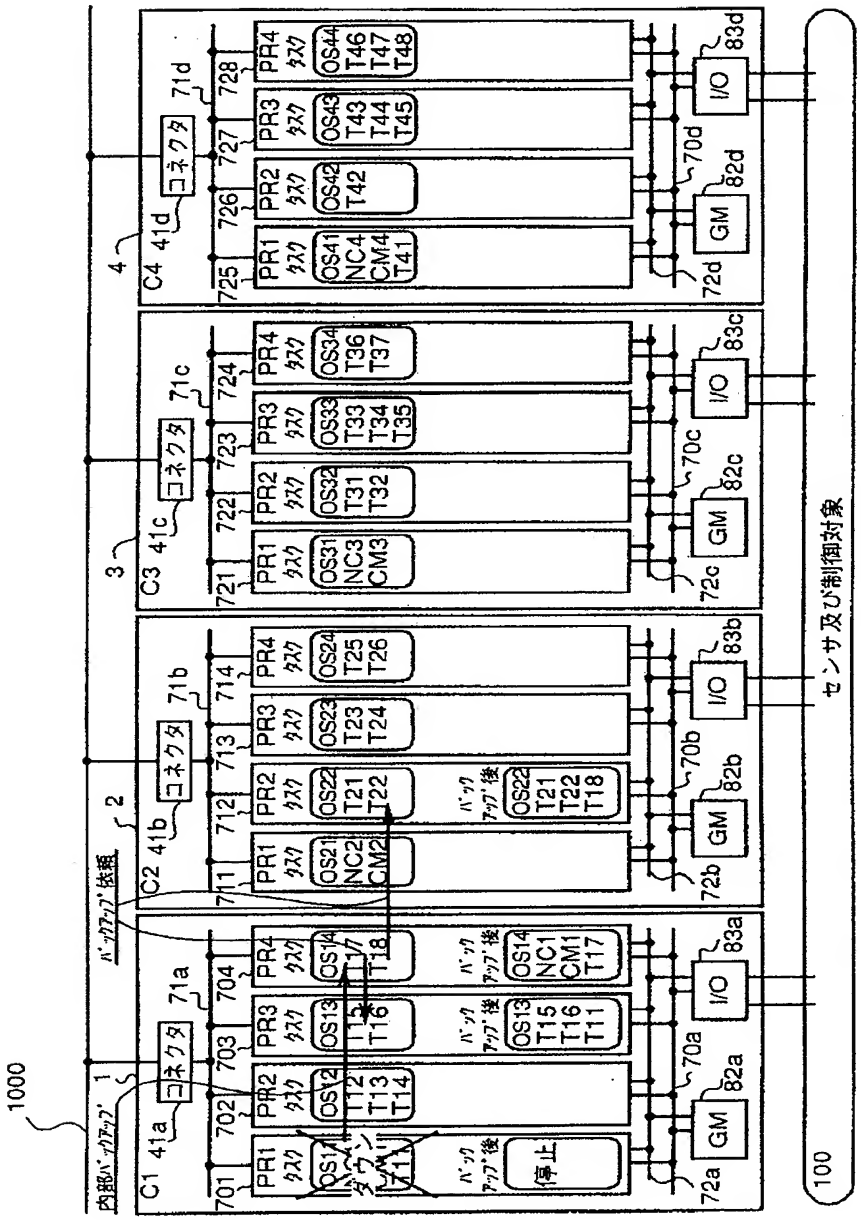
【図４７】

図　47


1000
DC1 41a
DC2 41b
DC3 41c
DC4 41d
ヨネクタ
71a
71b
71c
71d
PR1
PR2
PR3
PR4
OS11 NC1 CM1 T11
OS12 T12 T13 T14
OS13 T15 T16
OS14 T17 T18
OS21 NC2 CM2
OS22 T21 T22
OS23 T23 T24
OS24 T25 T26
OS31 NC3 CM3
OS32 T31 T32
OS33 T33 T34 T35
OS34 T36 T37
OS41 NC4 CM4 T41
OS42 T42
OS43 T43 T44 T45
OS44 T46 T47 T48
GM
I/O
センサ及び制御対象
100

【図５１】

図　51



フロントページの続き

(72)発明者　柴田　克成
東京都国分寺市東恋ケ窪一丁目280番地
株式会社日立製作所中央研究所内

(72)発明者　松原　清
東京都小平市上水本町五丁目20番１号　株式会社日立製作所半導体事業部内

(72)発明者　諸岡　泰男
茨城県日立市大みか町七丁目１番１号　株式会社日立製作所日立研究所内

(72)発明者　舩橋　誠壽
神奈川県川崎市麻生区王禅寺1099番地　株式会社日立製作所システム開発研究所内

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１１年（１９９９）１１月３０日

【公開番号】特開平８－２０２４０９
【公開日】平成８年（１９９６）８月９日
【年通号数】公開特許公報８－２０２５
【出願番号】特願平７－１２２９３
【国際特許分類第６版】
G05B 15/02
G06F 15/16 370
470
【ＦＩ】
G05B 15/02 M
G06F 15/16 370 Z
470 B

【手続補正書】
【提出日】平成１１年３月３１日
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】複数のプロセッサからなる制御システム用コントローラにおいて各コントローラ内の複数のプロセッサは自己の健全／異常の判別手段、プロセッサの負荷を測定または予測する手段を有し、一つのプロセッサが異常または負荷オーバーとなることを予測または検知した場合、他のプロセッサが負荷の代替を行うことを特徴とするコントローラ。
【請求項２】複数のコントローラと各コントローラを接続する伝送手段を有する分散制御システムにおいて各コントローラのうち少なくとも一台のコントローラはメモリまたは入出力を共有する複数のプロセッサから構成され、該伝送手段は該コントローラ内プロセッサ群の正常、異常に関係なく該コントローラの上記メモリまたは入出力の内容を読み書きできるように構成したことを特徴とする分散制御システム。
【請求項３】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの故障を検出する第１の手段と、各プロセッサの負荷状況を記憶する第２の手段と、各プロセッサの負荷を制御する第３の手段を有し、第１の手段が一つのプロセッサの故障を検出した場合に、第２の手段に記憶された各プロセッサの負荷状況に応じて第３の手段が故障したプロセッサのタスクを他の一つ以上のプロセッサに対して選択して代替させることを特徴とするコントローラ。
【請求項４】複数のプロセッサを有し、各プロセッサが複数のタスクを実行している制御システム用コントローラにおいて、各プロセッサの負荷を検出する第１の手段と、各プロセッサの負荷状況を記憶する第２の手段と、各プロセッサの負荷を制御する第３の手段を有し、第１の手段が一つのプロセッサの負荷オーバを検出した場合に、第２の手段に記憶された各プロセッサの負荷状況に基づいて、第３の手段が負荷オーバとなったプロセッサのタスクと他の一つ以上のプロセッサが実行しているタスクとからその一つ以上のプロセッサの各々が実行するタスクを選択し、それらプロセッサが選択されたタスクを実行することを特徴とするコントローラ。
【請求項５】複数のコントローラと各コントローラを接続するネットワークを有する分散制御システムにおいて、各コントローラは自コントローラの負荷を測定し測定した負荷をネットワークに送信する第１の手段と、任意のコントローラの負荷の代替を他のコントローラに依頼する第２の手段と、他のコントローラから代替の依頼に対して自己の負荷状況に基づき回答する手段をする第３の手段を有し、何れかのコントローラが負荷オーバあるいは故障した場合、その負荷オーバあるいは故障を検出したコントローラの第２の手段は各第１の手段から伝えられた負荷状況に応じて負荷の代替を依頼するコントローラを定め負荷の代替を依頼し、依頼されたコントローラの第３の手段は自己の負荷状況に応じて代替する負荷を定めこの負荷の実行を制御すると共にこの代替状況を回答することを特徴とした分散制御システム。
【請求項６】複数のコントローラを有する分散制御システムにおいて少なくとも一台のコントローラは複数のプロセッサから構成され、該複数のプロセッサ各々は自己の健全／異常の判別手段、各プロセッサの負荷を測定する手段を有し、一つのプロセッサが負荷オーバーになる

と、他のプロセッサのうち最も負荷の小さいプロセッサが負荷の代替を行うことを特徴とする分散制御システム。

【請求項７】複数のコントローラをネットワークで接続して制御を実行する分散制御システムにおいて、該複数のコントローラにおいて感知した異常をネットワーク上に送信するか否かの判定をネットワークからの受信情報に基づいて各コントローラで実行する分散型コントローラの診断方式。